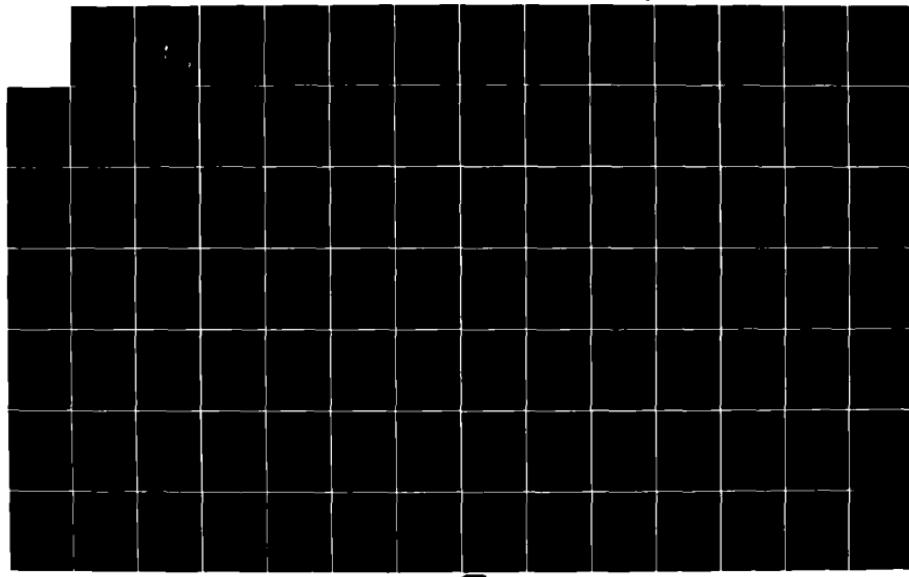


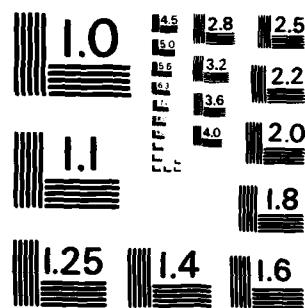
AD-A124 789 TACTICAL AIR CONTROL SYSTEM SIMULATION PROGRAM(U) AIR
FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF
ENGINEERING D P MCCAMLESS DEC 82 AFIT/OCS/EE/82D-24
F/G 9/2

UNCLASSIFIED

1/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

①

AD A 124 189



DTIC FILE COPY

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

DTIC
SELECTED
FEB 23 1983
S E D
E

Wright-Patterson Air Force Base, Ohio

This document has been approved

83 02 022063

AFIT/GCS/EE/82D-24

TACTICAL AIR CONTROL SYSTEM

SIMULATION PROGRAM

THESIS

AFIT/GCS/EE/82D-24

Donald P. McCanless
Captain USAF

Approved for public release; distribution unlimited.

TACTICAL AIR CONTROL SYSTEM
SIMULATION PROGRAM

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

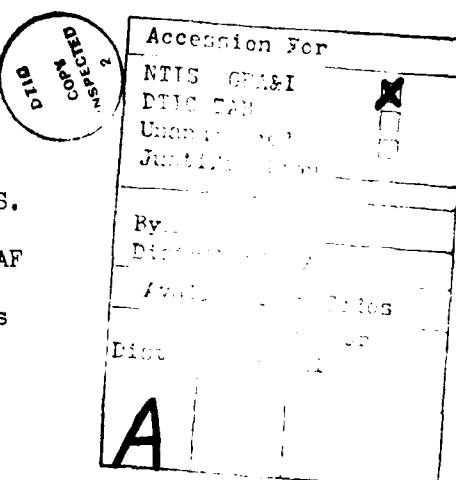
by

Donald P. McCanless, B. S.

Captain USAF

Graduate Computer Systems

December 1982



Approved for public release; distribution unlimited.

Acknowledgements

I would like to acknowledge the assistance provided by my thesis advisor, Lieutenant Colonel Harold Carter. Without his patience and guidance this effort would not have been completed. I would also like to thank my readers, Major Walter Seward and Dr. David Barr, for finding the time in their busy schedules to read this document and provide suggestions for improvement. Lastly, I would like to thank Professor Dan Reynolds for all of the enthusiastic support he provided.

Contents

	Page
Acknowledgements	ii
List of Figures.	iv
Abstract	v
I. Introduction	1
II. The Pre-Processor.	2
Introduction.	2
SLAMPP Capabilities	2
Nodes	2
Terminate Nodes	3
Message Creation.	3
Routing	3
Feasibility Study	3
Selected Capabilities	3
Design Stages	3
Program Evaluation.	4
Completed Design.	5
III. Program Installation	9
Introduction.	9
Array Size Selection.	9
Compilation	9
Enlarging SLAM.	10
IV. SLAMPP Validation.	11
Introduction.	11
Test Case	11
Test Case Validation.	11
Error Detection	12
V. Summary.	13
Bibliography	14
Appendix A: Program Listing	15
Appendix B: Operating Instructions.	51
Appendix C: Input Errors.	62
Appendix D: Interpreting the SLAM Output.	66
Appendix E: Test and Evaluation.	70

List of Figures

<u>Figure</u>		<u>Page</u>
1	Program Flow.	7
2	SLAM Main Routine	10

Abstract

The Tactical Air Force's Interoperability Group wants to be able to simulate communication networks. The simulation language for alternative modeling (SLAM) was proposed as the basis for the simulation. However, this language requires too much effort to make any changes to the network to be simulated. A Pre-Processor was written in FORTRAN 77 which allows the user to describe the network in a convenient manner, and produces the necessary SLAM source code to perform the simulation.

TACTICAL AIR CONTROL SYSTEM
SIMULATION PROGRAM

I. Introduction

The Tactical Air Force Interoperability Group (TAFIG) located at Langley Air Force Base, Virginia, wishes to be able to simulate the flow of message traffic through the Tactical Air Control System. This simulation will allow TAFIG to determine the number of messages going across particular communication channels and statistics about the queues which may form at nodes (Morrison, 1981). Current models, concluded one study, are too broad and it was recommended that a specific set of problems be defined and selected model development be continued (Bennett, 1970: ix).

A simulation language for alternative modeling (SLAM) was selected as the basis for the simulation. However, every change to the communication network being modeled would require rewriting the SLAM source code. This constraint was not seen as being "user-friendly" enough for a production system.

In order to make the ease of input acceptable, a SLAM Pre-Processor was proposed. This Pre-Processor would take, as inputs, a description of the network to be simulated along with descriptions of the message traffic and produce, as output, the necessary SLAM source code to model the network. This SLAM source code would, in turn, be input into the SLAM program for the actual simulation.

II. The Pre-Processor

Introduction

In order to determine the feasibility of such a system as the SLAM Pre-Processor (SLAMPP), a subset of the desired capabilities was selected and implemented. Following this evaluation, the system was expanded to include all of the desired capabilities.

SLAMPP Capabilities

The SLAMPP was designed to allow the user to simulate communication networks. To the SLAMPP, a network consists of message creations and queues (processing locations), along with the path a message will follow through the various queues.

In order to describe a communication network, SLAMPP must allow the user to describe the network to be simulated. The items which may be described are listed below.

Nodes. A node, or queue, is a location at which a message will receive some type of handling. The maximum number of messages which may be waiting at a node at any given time can be specified. The name of an alternate node may (optionally) be supplied to which a message will "balk" should the node's capacity be exceeded, or "blocking" may be specified which causes messages to "back up" in the system when capacity is exceeded. If neither balking nor blocking is specified, the message will simply be lost (without any indication to the user) if capacity has been reached.

Two time durations are associated with a node. The first allows

the user to specify how long it takes to process a message at this node, while the second specifies how long transmission to the next node will take. If either of these two times are not specified, a default time of zero will be assumed.

Terminate Nodes. Terminate nodes are locations at which a message leaves the network.

Message Creation. The rate at which a message of a particular type is created can be specified. Additionally, the starting time for this message and the number of messages to be created are also specified.

Routing. Each message may be routed through the nodes following paths which the user supplies. Also, an alternate balk route can be supplied should a message be redirected to a balk node.

Feasibility Study

In order to determine the feasibility of the SLAMPP, a subset of the capabilities was selected and implemented.

Selected Capabilities. Only the "bare" essential capabilities were selected for the feasibility study. Additionally, these capabilities were (in some cases) simplified.

Node definition was included; however, balk nodes were not implemented. Terminate nodes were included exactly as they would be included in the final version, as was message creation. Path definition did not support balk paths, nor blocking. Message processing times and transmit times between nodes were ignored.

Design Stages. Following the selection of the capabilities to be studied, the feasibility fell into several stages. First, a logical and easy-to-follow method was developed to input the network description to

the Pre-Processor. These input formats, with some modifications and additions, remained in the final version.

Next, a small network was designed and, using the input formats mentioned previously, a SLAMPP description of this network was coded.

A great deal of information was necessary to build even this simple implementation. Storing and manipulating this data quickly became confusing. In order to try and clarify this problem, the variables and data structures to store this information was designed. Although some modifications occurred during the course of the project, the basic design remained the same.

Now that the exact capabilities for the feasibility study were known, and the inputs and data structures were designed, flowcharts were produced which showed the steps necessary to implement the project. The flowcharts began at a very "high-level" and were iteratively refined until the detail necessary to design the program was obtained.

Using these flowcharts, the FORTRAN source code was written. Although some logic errors did exist, they proved to be relatively minor and easy to correct.

Program Evaluation. After the SLAMPP was written and debugged, the Pre-Processor output was examined for correctness. This examination included a visual inspection of the output file to verify the SLAM code produced by the Pre-Processor was as expected, and an execution of this code by the SLAM program to ensure the code would indeed be accepted by the SLAM system.

This examination showed that the SLAMPP output was valid and acceptable to the SLAM program. Additionally, it showed that the basic

concept behind the Pre-Processor was viable and continued development was warranted.

Completed Design

Following the feasibility study, the remaining capabilities were incorporated into the final SLAMPP. Due to modularity of design these additions were easily installed into the framework developed earlier (see Appendix A for a listing of the SLAMPP program).

The basic flow of the program is shown in Figure 1, while Appendix B contains detailed operating instructions. First, all of the nodes are defined, followed by all of the terminate nodes. After these definitions comes the start of the first message. Each message consists of one message definition, followed by an optional process and transmit time definition, followed by the name of the next node along the path. Successive steps in the path are indicated by repetition of process and transmit times and node name. Additional messages are input by placing another message definition into the data deck and repeating the above steps.

After all messages have been described, information about balk routes (if any are present) is input. This process is similar to message definition, however, the message definition card is replaced by a balk route definition card. The process and transmit time and path definitions are the same as above. One balk definition is included for each message that can possibly balk.

The validity of the inputs is verified prior to building the SLAM source code (a summary of the input errors which can be detected is

contained in Appendix C). If all inputs are valid, the SLAM source is constructed.

The SLAM source built by the Pre-Processor now serves as input to the SLAM program for the actual simulation. The outputs produced by the SLAM program are described in Appendix D.

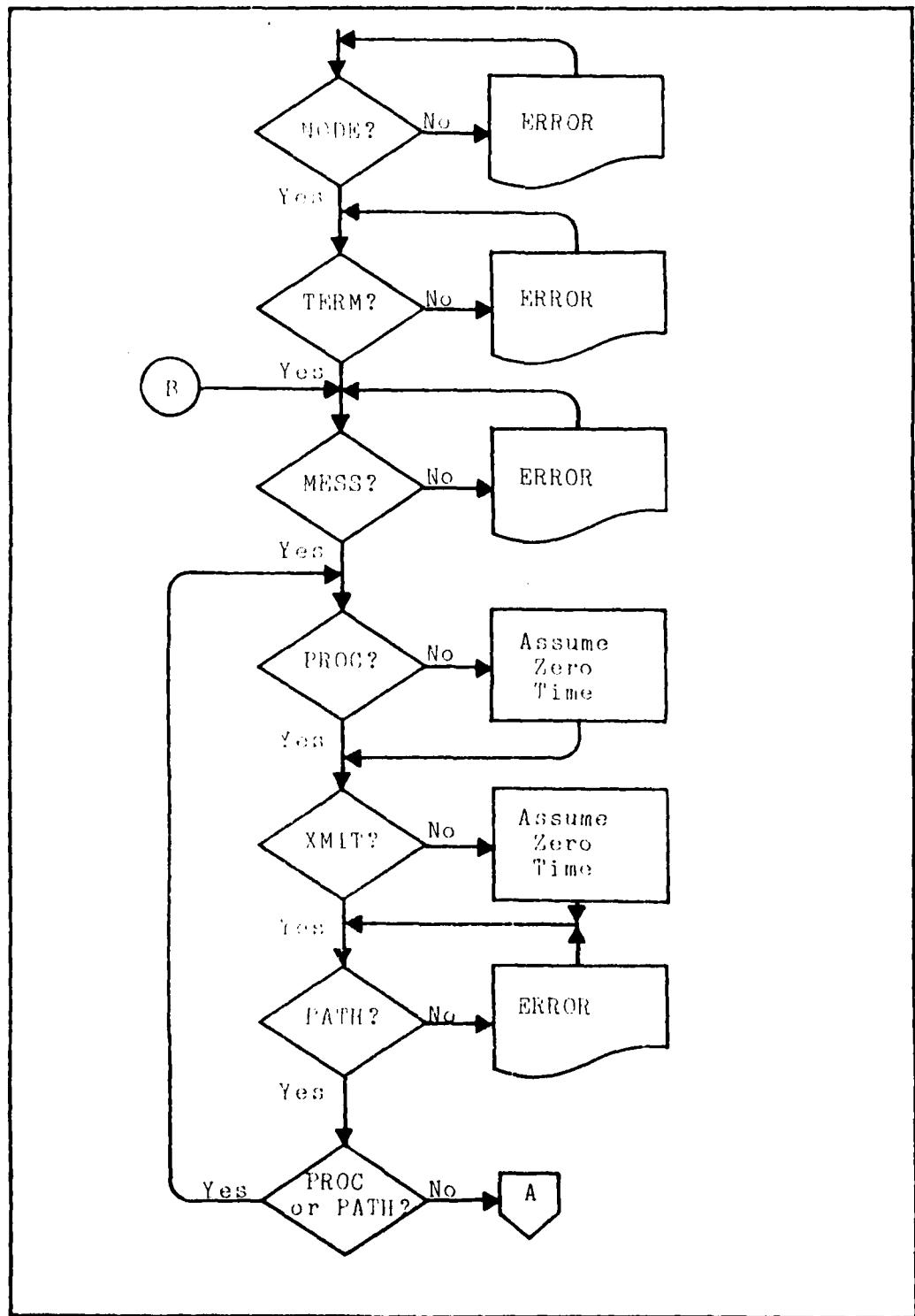


Figure 1. Program Flow (Part One)

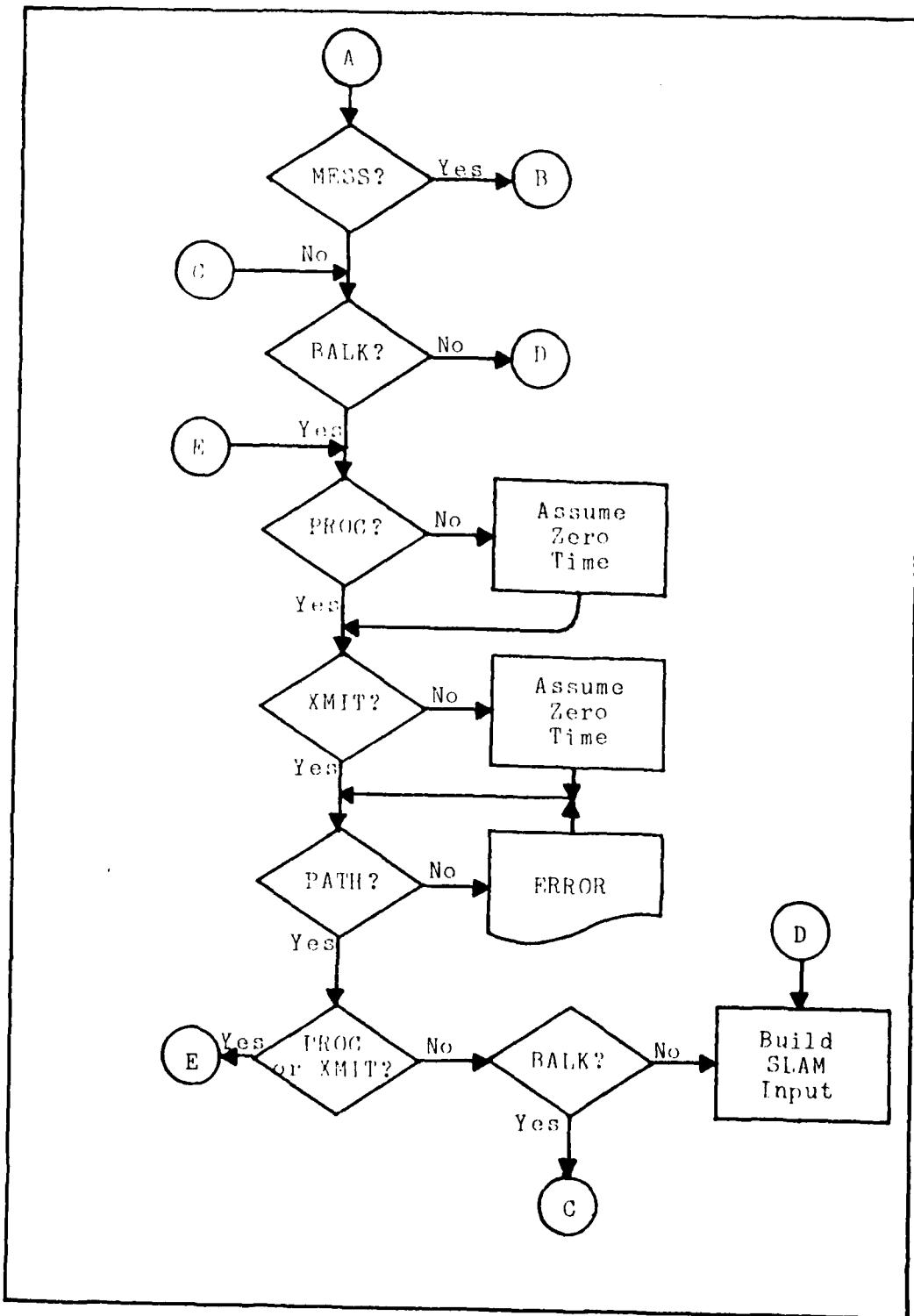


Figure 1. Program Flow (Part Two)

III. Program Installation

Introduction

Prior to execution of the SLAM Pre-Processor, the program must be compiled. To minimize the amount of computer memory required, certain changes can be made to the program to decrease array sizes.

Array Size Selection

Due to the way SLAMPP is written, the user has the flexibility to vary the maximum number of messages (NOMSG) the Pre-Processor can handle, the maximum number of terminate nodes (NOTRM), and the maximum number of nodes (NONOD).

SLAMPP is written entirely in FORTRAN 77. To allow for variable dimension sizes for arrays, each subroutine and function contains a PARAMETER statement (Katzan, 1978: 77) which defines each of the three variables sizes. Prior to compilation, the user should decide on appropriate values for these parameters and change each PARAMETER statement accordingly.

Since each of the three parameters are used to determine array sizes, the larger the parameter sizes, the larger the amount of computer memory required to execute SLAMPP. Therefore, care should be used to select values which are both realistic and conservative.

Compilation

As mentioned previously, SLAMPP is written in FORTRAN 77. After any changes to the PARAMETER statements are made, the Pre-Processor is

ready to be compiled and saved using whatever installation dependent procedures are required.

Enlarging SLAM

Figure 2 shows the main driver for the SLAM program. Depending upon the size of the SLAMPP network to be simulated, certain changes may be necessary to the SLAM main routine to accommodate the network. Although these changes are part of the SLAM program itself, and not part of the Pre-Processor, they are listed here for the user's convenience.

Three items must be changed: the dimension sizes of NSET and QSET, and the value for NNSET. Each of these three values must be the same to ensure a successful SLAM run. Typically, the default value is 5,000.

```
PROGRAM MAINRT (INPUT, OUTPUT, TAPE7, TAPE5=INPUT,
& TAPE6=OUTPUT)
C
C THIS ROUTINE IS INCLUDED SO THAT THE ARRAY SIZE
C FOR NNSET/QSET CAN BE CHANGED TO ACCOMODATE
C LARGE SIMULATION RUNS.
C
DIMENSION NSET(7000)
COMMON /SCOM1/ ATRIB(100), DD(100), DDL(100),
& DTNOW, II, MFA, MSTOP, NCLNR, NCRDR, NPRNT,
& TNEXT, TNOW, XX(100)
COMMON QSET(7000)
EQUIVALENCE (NSET(1), QSET(1))
NNSET = 7000
NCRDR = 5
NPRNT = 6
NTAPE = 7
CALL SLAM
STOP
END
```

Figure 2. SLAM Main Driver

IV. SLAMPP Validation

Introduction

In order to check the validity of the SLAMPP (that is, verify that SLAMPP produces a SLAM network which represents the intention of the inputs), a test case was designed and evaluated. In addition to this large test case, over 30 smaller networks were input to the Pre-Processor during development to assure a valid Pre-Processor was in fact produced. Also, tests were performed to ensure that invalid inputs would be detected.

Test Case

The test case used for validation was designed to exercise every capability of SLAMPP. Specifically, nodes of all three types were used (some with balking, some with blocking, and some with neither). Nine different messages were included to demonstrate each of the nine valid distribution types (see Appendix B). Process and transit times included at least one example of each of the nine distributions also, plus examples of the zero default time which will be generated if not specified otherwise. Since balking was specified on two nodes, balk route descriptions were required in addition to regular route descriptions. See Appendix E for a listing of the SLAMPP inputs and the associated SLAM source which was generated.

Test Case Validation

After the test network was designed, the SLAMPP description of the

network was written and input into the Pre-Processor. The SLAM source output was then carefully examined to ensure that a valid network representation had been produced. Also, the Pre-Processor output was executed through the SLAM program to verify it would execute successfully. Both the examination and the execution showed the SLAMPP output to contain no known discrepancies.

Error Detection

Given valid inputs, SLAMPP will produce a corresponding SLAM network; however, it is equally important that the Pre-Processor recognize invalid inputs to prevent the creation of a bad network.

Many types of input validations are performed by the Pre-Processor. The errors which may be detected are summarized in Appendix C. To be certain that each input validation point worked correctly, test cases were devised which intentionally (and unintentionally) exercised each point. Since the validity of any SLAM network created using invalid input data is questionable, no SLAM source is generated if any errors are detected.

V. Summary

The Tactical Air Forces Interoperability Group needs to be able to simulate the flow of message traffic through the Tactical Air Control System. The simulation language for alternative modeling (SLAM) was used as the basis for the simulation. A Pre-Processor was constructed to permit a user-friendly operation.

The Pre-Processor was written in FORTRAN 77, and produced the source code to be used by the SLAM program. The Pre-Processor output was validated and found to be correct.

Bibliography

Bennett, A. S., et al. CONSTANT QUEST Modeling Group: Phase I Report.
Bedford, Mass.: The Mitre Corp., 2 April 1979.

Katzan, Harry Jr., FORTRAN 77. New York: Van Nostrand Reinhold, 1978.

Morrison, John. Captain. "Proposed AFIT Thesis Topic," Tactical
Interoperability Group, Langley Air Force Base, Virg.

Pritsker, A. Alan B. and Claude Dennis Pegden. Introduction to
Simulation and SLAM. New York: Halsted Press, 1979.

Appendix A: Program Listing

Following is the FORTRAN 77 listing of the source code for the SLAM
Pre-Processor.

PROGRAM SLAMPP

C THIS PROGRAM ACCEPTS (FROM FILE 10) INPUT CARDS DESCRIBING THE
C NETWORK TO BE SIMULATED, CHECKS THEIR VALIDITY, AND (IF NO
C ERRORS ARE DETECTED) PRODUCES THE CORRESPONDING SLAM SOURCE
C CODE ON FILE 20.
C
C SUBROUTINES CALLED:
C BLDOUT--BUILD THE SLAM OUTPUT
C CHKBLK--VALIDATE BALK ROUTES
C CHKUSD--ENSURE ALL NODES ARE USED
C BALK--PROCESS BALK CARDS
C MESS--PROCESS MESS CARDS
C NODE--PROCESS NODE CARDS
C PROC--PROCESS PROC CARDS
C PATH--PROCESS PATH CARDS
C TERM--PROCESS TERM CARDS
C XMIT--PROCESS XMIT CARDS
C READ--READ INPUT FILE
C
C FUNCTIONS CALLED:
C NUMNOD--RETURNS THE NODE'S NUMBER
C NUMTRM--RETURNS THE TERM'S NUMBER
C
C INPUT ARGUMENTS:
C NONE
C
C OUTPUT ARGUMENTS:
C NONE
C
C INPUTS VIA COMMON:
C IFATAL
C NXTNOD
C NXTTRM
C NXTMSG
C PARM1
C PARM2
C PARM3
C PARM4
C PARM5
C CRDTYP
C LABEL
C LABEL2
C DSTNAM
C
C OUTPUTS VIA COMMON:
C IFATAL
C NXTNOD
C NXTTRM
C NXTMSG
C MAXNOD
C MAXTRM
C MAXMSG
C PTHMAT
C NODNAM
C TRMNAM
C DSTNAM
C
C WORKING VARIABLES:
C I--DO LOOP COUNTER

```

C      ISAVMS--TEMPORARY STORAGE OF MESSAGE COUNT
C      ISAVND--TEMPORARY STORAGE OF NODE COUNT
C      J--DO LOOP COUNTER
C
C
C      PARAMETERS FOR DIMENSION SIZES:
C      NONOD--MAXIMUM NUMBER OF NODES
C      NOTRM--MAXIMUM NUMBER OF TERMINATE NODES
C      NOMSG--MAXIMUM NUMBER OF MESSAGES
C
C      PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
C      CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4
C      & BALKS*4, DSTNAM*5
C      DESCRIPTION OF COMMON BLOCK /COM1/:
C
C      IFATAL--NUMBER OF FATAL ERRORS DETECTED
C      NXTNOD--CURRENT NODE NUMBER
C      NXTRM--CURRENT TERMINATE NODE NUMBER
C      NXTMSC--CURRENT MESSAGE NUMBER
C      MAXNOD--MAXIMUM NUMBER OF NODES
C      MAXTRM--MAXIMUM NUMBER OF TERMINATE NODES
C      MAXMSG--MAXIMUM NUMBER OF MESSAGES
C      NODCAP(1,I)--MAXIMUM CAPACITY OF NODE I
C      NODCAP(2,I)--NUMBER OF SERVERS AT NODE I
C      PTHMAT(I,J)--
C          IF > 0, NODE NUMBER OF NEXT NODE THAT MESSAGE J
C              WILL FOLLOW AFTER NODE I
C          IF = 0, MESSAGE J DOES NOT USE NODE I
C          IF < 0, NEGATIVE OF THE TERMINATE NODE NUMBER THAT
C              MESSAGE J WILL FOLLOW AFTER NODE I
C      CREATE(1,I)--PARAMETER 1 OF DISTRIBUTION RATE FOR
C          MESSAGE I
C      CREATE(2,I)--PARAMETER 2 OF DISTRIBUTION RATE FOR
C          MESSAGE I
C      CREATE(3,I)--PARAMETER 3 OF DISTRIBUTION RATE FOR
C          MESSAGE I
C      CREATE(4,I)--STARTING TIME FOR MESSAGE I
C      CREATE(5,I)--NUMBER OF MESSAGE RELEASES
C      LASNOD(I)--THE PREVIOUS NODE NUMBER IN THE PATH FOR
C          MESSAGE I
C      NODSTR(I)--THE FIRST NODE NUMBER IN THE PATH FOR
C          MESSAGE I
C      PARM1--1ST NUMERIC DATA FIELD READ FROM INPUT DECK
C      PARM2--2ND NUMERIC DATA FIELD READ FROM INPUT DECK
C      PARM3--3RD NUMERIC DATA FIELD READ FROM INPUT DECK
C      PARM4--4TH NUMERIC DATA FIELD READ FROM INPUT DECK
C      PARM5--5TH NUMERIC DATA FIELD READ FROM INPUT DECK
C      DSTPRC(I,J,1)--DISTRIBUTION NUMBER FOR THE PROCESSING
C          TIME FOLLOWING NODE J FROM MESSAGE I
C      DSTPRC(I,J,2)--PARAMETER 1 OF PROCESSING DISTRIBUTION
C      DSTPRC(I,J,3)--PARAMETER 2 OF PROCESSING DISTRIBUTION
C      DSTPRC(I,J,4)--PARAMETER 3 OF PROCESSING DISTRIBUTION
C      DSTXMT(I,J,1)--DISTRIBUTION NUMBER FOR THE TRANSMIT
C          TIME FOLLOWING NODE J FROM MESSAGE I

```

```

C      DSTXMT(I,J,2)--PARAMETER 1 OF TRANSMIT DISTRIBUTION
C      DSTXMT(I,J,4)--PARAMETER 2 OF TRANSMIT DISTRIBUTION
C      DSTXMT(I,J,4)--PARAMETER 3 OF TRANSMIT DISTRIBUTION
C
C      COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
C      & MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
C      & CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
C      & PARM1, PARM2, PARM3, PARM4, PARM5,
C      & DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C      DESCRIPTION OF COMMON BLOCK /COM2/:
C
C      NODNAM(I)--NAME OF NODE I
C      DIST(I)--NAME OF THE MESSAGE DISTRIBUTION FOR MESSAGE I
C      TRMNAM(I)--NAME OF TERMINATE NODE I
C      CRDTYP--CARD TYPE READ FROM INPUT DECK
C      LABEL--1ST CHARACTER DATA FIELD READ FROM INPUT DECK
C      LABEL2--2ND CHARACTER DATA FIELD READ FROM INPUT DECK
C      BALKS(I)--
C          IF BLANK, NODE I DOES NOT BLOCK OR BALK
C          IF 'BLOK', NODE I IS BLOCKED
C          OTHERWISE, NODE NAME THAT NODE I BALKS TO
C      DSTNAM--LIST OF VALID DISTRIBUTION NAMES
C
C      COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
C      & LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C      PREPARE INPUT FILE
C      REWIND 10
C      INITIAL VALUES
C          IFATAL = 0
C          NXTNOD = 0
C          NXTTRM = 0
C          NXTMSG = 0
C          MAXNOD = NONOD
C          MAXTRM = NOTRM
C          MAXMSG = NOMSG
C      VALID DISTRIBUTION NAMES
C          DSTNAM(1) = 'EXPO'
C          DSTNAM(2) = 'NPSSN'
C          DSTNAM(3) = 'BETA '
C          DSTNAM(4) = 'GAMA '
C          DSTNAM(5) = 'RLOGN'
C          DSTNAM(6) = 'RNORM'
C          DSTNAM(7) = 'UNFRM'
C          DSTNAM(8) = 'TRIAG'
C          DSTNAM(9) = 'CONST'
C          DO 1 I = 1, MAXMSG
C          DO 1 J = 1, MAXNOD
C      ZERO OUT PTHMAT (I.E., ESTABLISH NO PATHS EXIST FOR ANY
C      MESSAGE).
1      PTHMAT(J, I) = 0.0
DO 2, I = 1, MAXTRM
C      INITIALIZE ALL TERMINATE NODES TO BLANK (UNUSED).
2      TRMNAM(I) = ' '
DO 3, I = 1, MAXNOD

```



```

C  EACH MESSAGE CONSISTS OF A MESS CARD, FOLLOWED BY A PROC (OPTIONAL),
C  AN XMIT (OPTIONAL), AND A PATH CARD. CONTINUE PROCESSING MESSAGES
C  UNTIL A NON-MESS, PROC, XMIT OR PATH CARD IS FOUND.
C
C  FOLLOWING TERM CARDS MUST BE A MESS CARD.
50  IF (CRDTYP .NE. 'MESS') THEN
C      SOMETHING OTHER THAN A MESS CARD WAS FOUND.
      PRINT *, 'ERROR--MESS CARD MUST FOLLOW TERM CARDS.'
      IFATAL = IFATAL + 1
      CALL READ
      GO TO 50
    ELSE
C      A MESS CARD WAS FOUND--PROCESS IT.
60  CALL MESS(LABEL, PARM1, PARM2, PARM3, PARM4, PARM5, LABEL2)
      CALL READ
C      FOLLOWING A MESS CARD CAN BE A PATH CARD, A PROC
C      CARD, OR AN XMIT CARD.
      IF (CRDTYP .EQ. 'PROC') THEN
C          A PROC CARD WAS FOUND--PROCESS IT.
          CALL PROC (LABEL, PARM3, PARM4, PARM1)
          CALL READ
        ELSE
C          NO PROC CARD WAS FOUND--ESTABLISH DEFAULT PROCESS TIME
          CALL PROC('CONST', 0.0, 0.0, 0.0)
        ENDIF
        IF (CRDTYP .EQ. 'XMIT') THEN
C          AN XMIT CARD WAS FOUND--PROCESS IT.
          CALL XMIT(LABEL, PARM3, PARM4, PARM1)
          CALL READ
        ELSE
C          NO XMIT CARD WAS FOUND--ESTABLISH DEFAULT TRANSMIT TIME
          CALL XMIT('CONST', 0.0, 0.0, 0.0)
        ENDIF
C          A PATH CARD MUST BE NEXT.
        IF (CRDTYP .NE. 'PATH') THEN
C          NO PATH CARD FOUND.
          NXTMSG = NXTMSG - 1
          PRINT *, 'ERROR--MISSING PATH CARD. SKIPPING TO NEXT MESS.'
70  CALL READ
C          GO TO BEGINNING OF NEXT MESSAGE.
          IF (CRDTYP .EQ. 'MESS') THEN
            GO TO 60
          ELSE
            GO TO 70
          ENDIF
        ELSE
C          A PATH CARD WAS FOUND--PROCESS IT.
80  CALL PATH
C          IF NUMTRM(LABEL) = 0, THEN THE LAST NODE SPECIFIED
C          WAS NOT A TERMINATE NODE. THEREFORE, THE NEXT CARD
C          MUST BE EITHER PROC, XMIT OR PATH TO CONTINUE
C          DESCRIBING THIS MESSAGE PATH.
          IF (NUMTRM(LABEL) .EQ. 0) THEN
C              LAST NODE WAS NOT A TERMINATE NODE.

```

```

CALL READ
IF (CRDTYP .EQ. 'PROC') THEN
    A PROC CARD WAS FOUND--PROCESS IT.
    CALL PROC(LABEL, PARM3, PARM4, PARM1)
    CALL READ
ELSE
    NO PROC CARD WAS FOUND--ESTABLISH DEFAULT PROCESS TIME.
    CALL PROC('CONST', 0.0, 0.0, 0.0)
ENDIF
IF (CRDTYP .EQ. 'XMIT') THEN
    AN XMIT CARD WAS FOUND--PROCESS IT.
    CALL XMIT(LABEL, PARM3, PARM4, PARM1)
    CALL READ
ELSE
    NO XMIT CARD WAS FOUND--ESTABLISH DEFAULT TRANSMIT
    TIME.
    CALL XMIT('CONST', 0.0, 0.0, 0.0)
ENDIF
IF (CRDTYP .EQ. 'PATH') THEN
    A PATH CARD WAS FOUND--PROCESS IT.
    GO TO 80
ENDIF
NO PATH CARD WAS FOUND.
PRINT *, 'ERROR--MESSAGE MUST END ON A TERMINATE NODE.'
IFATAL = IFATAL + 1
ELSE
    THE LAST NODE WAS A TERMINATE NODE.
    CALL READ
ENDIF
ENDIF
FOLLOWING THE LAST PATH CARD OF A MESSAGE MAY BE ANOTHER
MESS CARD, A BALK CARD, OR AN END CARD.
IF (CRDTYP .EQ. 'MESS') THEN
    ANOTHER MESSAGE IS BEING DEFINED.
    GO TO 50
-----
ENDIF
SAVE NXTNOD AND NXTMSG FOR LATER USE.
ISAVND = NXTNOD
ISAVMS = NXTMSG
IF (CRDTYP .NE. 'BALK') THEN
    A BALK ROUTE IS NOT BEING DEFINED.
    GO TO 500
ENDIF
----- PROCESS BALK ROUTES -----
EACH BALK ROUTE CONSISTS OF A BALK CARD, FOLLOWED BY A PROC(OPTIONAL)
AN XMIT (OPTIONAL), AND A PATH CARD.  CONTINUE PROCESSING BALK ROUTE
UNTIL A NON-BALK, PROC, XMIT ORPATH CARD IS FOUND

```

```

C      ESTABLISH THE NXTNOD NUMBER FOR THIS BALK ROUTE.
400  NXTNOD = NUMNOD(LABEL)
      IF (NXTNOD .EQ. 0) THEN
          PRINT *, 'ERROR--INVALID BALK NODE (SKIPPING TO NEXT BALK.)'
          PRINT *, '          CORRECT NODE NAME OR INSERT CORRESPONDING',
          &           'TERM CARD.'
          IFATAL = IFATAL + 1
402  CALL READ
      IF (CRDTYP .EQ. 'BALK') THEN
          GO TO 400
      ELSE
          GO TO 402
      ENDIF
  ENDIF
C      ESTABLISH THE NXTMSG NUMBER FOR THIS BALK ROUTE.
C      NXTMSG IS THE MESSAGE NUMBER OF THE MESSAGE FOR WHOM A BALK
C      ROUTE IS BEING DEFINED.
      NXTMSG = PARM1
      IF (NXTMSG .GT. ISAVMS .OR. NXTMSG .LE. 0) THEN
          PRINT *, 'ERROR--INVALID MESSAGE NUMBER (SKIPPING TO NEXT ',
          &           'BALK).'
          IFATAL = IFATAL + 1
401  CALL READ
      IF (CRDTYP .EQ. 'BALK') THEN
          GO TO 400
      ELSE
          GO TO 401
      ENDIF
  ENDIF
C      A VALID BALK CARD WAS FOUND--PROCESS IT.
      CALL BALK
      CALL READ
C      FOLLOWING A BALK CARD MAY BE EITHER A PROC CARD, XMIT
C      CARD, OR PATH CARD.
410  IF (CRDTYP .EQ. 'PROC') THEN
      C      A PROC CARD WAS FOUND--PROCESS IT.
          CALL PROC(LABEL, PARM3, PARM4, PARM1)
          CALL READ
      ELSE
          NO PROC CARD WAS FOUND--ESTABLISH DEFAULT PROCESS TIME.
          CALL PROC ('CONST', 0.0, 0.0, 0.0)
      ENDIF
      IF (CRDTYP .EQ. 'XMIT') THEN
          C      AN XMIT CARD WAS FOUND--PROCESS IT.
          CALL XMIT(LABEL, PARM3, PARM4, PARM1)
          CALL READ
      ELSE
          NO XMIT CARD WAS FOUND--ESTABLISH DEFAULT TRANSMIT TIME.
          CALL XMIT('CONST', 0.0, 0.0, 0.0)
      ENDIF
      IF (CRDTYP .NE. 'PATH') THEN
          C      THIS BALK ROUTE CONTAINED NO PATH.
          PRINT *, 'ERROR--MISSING BALK PATH CARD. SKIPPING TO NEXT ',
          &           'BALK.'

```


6 'CONSTRUCTED.'
C
C - - - - -
C
ENDIF
RETURN
END

```

SUBROUTINE BALK

C THIS SUBROUTINE HANDLES BALK CARDS. WHEN A BALK IS ENCOUNTERED,
C LASNOD (THE PREVIOUS NODE NUMBER) IS INITIALIZED TO THE FIRST
C NODE IN THE BALK PATH.

C CALLED BY:
C     MAIN
C SUBROUTINES CALLED:
C     NONE
C FUNCTIONS CALLED:
C     NUMNOD
C INPUT ARGUMENTS:
C     NONE
C OUTPUT ARGUMENTS:
C     NONE
C INPUTS VIA COMMON:
C     IFATAL
C     NXTMSG
C     LABEL
C OUTPUT VIA COMMON:
C     IFATAL
C     LASNOD
C WORKING VARIABLES:
C     NUMN--NODE NUMBER
C
PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4
& BALKS*4, DSTNAM*5
COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
& MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
& CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
& PARM1, PARM2, PARM3, PARM4, PARM5,
& DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
& LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C FIND THE NODE NUMBER OF THE BALK NODE (NUMN MUST BE > 0 TO
C BE VALID).
NUMN = NUMNOD(LABEL)
IF (NUMN .GT. 0) THEN
    INITIALIZE LASNOD TO THE 1ST NODE IN THIS BALK PATH.
    LASNOD(NXTMSG) = NUMN
ELSE
    PRINT *, 'ERROR--ILLEGAL BALK NODE NAME.'
    IFATAL = IFATAL + 1
ENDIF
RETURN
END

```

```

SUBROUTINE BLDOUT
C
C      THIS SUBROUTINE PRODUCES THE SLAM SOURCE CODE, WHICH IS
C      WRITTEN TO FILE 20.
C
C      CALLED BY:
C          MAIN
C
C      SUBROUTINES CALLED:
C          NONE
C
C      FUNCTIONS CALLED:
C          NUMDST
C
C      INPUT ARGUMENTS:
C          NONE
C
C      OUTPUT ARGUMENTS:
C          NONE
C
C      INPUTS VIA COMMON:
C          NXTNOD
C          NXTTRM
C          NXTMSG
C          NODCAP
C          PTHMAT
C          CREATE
C          NODSTR
C          DSTPRC
C          DSTXMT
C          NODNAM
C          DIST
C          TRMNAM
C          BALKS
C          DSTNAM
C
C      OUTPUTS VIA COMMON:
C          NONE
C
C      WORKING VARIABLES:
C          BRNAME--BRANCH NODE NAME
C          I--DO LOOP COUNTER
C          ICNT--GENERATED LINE LABEL NUMBER
C          IMAX--INTEGER REPRESENTATION OF STARTING TIME OF MESSAGE
C          IMIN--INTEGER REPRESENTATION OF NUMBER OF MESSAGE RELEASES
C          ISTRT--LINE LABEL NUMBER
C          J--DO LOOP COUNTER
C          NUM--THE NEXT NODE NUMBER IN THE PATH
C          NUMD--DISTRIBUTION NUMBER
C
C
PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
CHARACTER BRNAME*4
CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4
& BALKS*4, DSTNAM*5
COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
& MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
& CREATE(6, NOMSG), LASHOD(NOMSG), NODSTR(NOMSG),
& PARM1, PARM2, PARM3, PARM4, PARM5,
& DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
& LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)

```

```

REWIND 20
C      WRITE SLAM GEN CARD.
      WRITE (20, 10)
10     FORMAT ('GEN,SLAMPP,SLAM,09/19/1982,1,,,,72;')
      WRITE (20, 15) NXTNOD
C      WRITE LIM CARD.
15     FORMAT ('LIM,',I4,',2,500;')
      WRITE (20,20)
20     FORMAT ('NETWORK;')
C      WRITE RESOURCE CARD FOR EACH NODE.
      DO 300, I = 1, NXTNOD
300   WRITE (20, 301) NODNAM(I), NODCAP(2, I), I
301   FORMAT ('      RESOURCE', A4, '(', I4, ',' , I4, ';')
C      WRITE MESSAGE CREATION CARDS.  EACH MESSAGE CONSISTS OF A
C      CREATE CARD, ASSIGN CARD, AND AN ACT CARD.
      DO 900 I = 1, NXTMSG
C      STARTING TIME FOR THIS MESSAGE.
      IMIN = CREATE (4, I)
C      NUMBER OF MESSAGE RELEASES.
      IMAX = CREATE (5, I)
C      NUMBER OF THE CREATION DISTRIBUTION RATE.
      NUMD = NUMDST(DIST(I))
C      EACH DISTRIBUTION RATE REQUIRES A CERTAIN NUMBER OF INPUT
C      PARAMETERS.  THE FOLLOWING SELECTS THE APPROPRIATE WRITE
C      STATEMENT FOR THE DISTRIBUTION.
      IF (NUMD .EQ. 9) THEN
C          CONSTANT CREATION RATE.
          WRITE (20, 800) CREATE(1, I), IMIN, IMAX
800   FORMAT ('      CREATE,', F10.5, ',', I5, ',1,', ,
&           I5, ';')
      ELSE IF (NUMD .LE. 2) THEN
C          FOR EXPON AND NPSSN (REQUIRING ONE PARAMETER).
          WRITE (20, 805) DIST(I), CREATE(1, I), IMIN, IMAX
805   FORMAT ('      CREATE,', A5, '(', F10.5, ',' , ,
&           I5, ',1,', I5, ';')
      ELSE IF (NUMD .LE. 7) THEN
C          FOR BETA, GAMMA, RLOGN, RNORM AND UNFRM (REQUIRING TWO
C          PARAMETERS).
          WRITE (20, 810) DIST(I), CREATE(1, I), CREATE(2, I),
&           IMIN, IMAX
810   FORMAT ('      CREATE,', A5, '(', F10.5, ',', F10.5,
&           '),', I5, ',1,', I5, ';')
      ELSE
C          FOR TRIAG (REQUIRING THREE PARAMETERS).
          WRITE (20, 820) DIST(I), CREATE(1, I), CREATE(2, I),
&           CREATE(3, I), IMIN, IMAX
820   FORMAT ('      CREATE,', A5, '(', 2(F10.5, ',' ,
&           F10.5, '),', I5, ',1,', I5, ';')
      ENDIF
C      EACH MESSAGE IS ASSIGNED A SEQUENTIAL MESSAGE NUMBER SO
C      THAT DIFFERENT MESSAGES CAN BE DISTINGUISHED FROM EACH
C      OTHER THROUGHOUT THE NETWORK.
      WRITE (20, 30) I
30     FORMAT ('      ASSIGN,ATRIB(2)=' , I4, ';')

```

```

C      ACT TO BRANCH TO 1ST NODE IN THE PATH.
      WRITE (20, 35) NODNAM(NODSTR(I))
35      FORMAT ('      ACT,,,',A4,';')
900      CONTINUE
C      ESTABLISH BEGINNING LABEL NUMBER FOR PROGRAM GENERATED LABELS.
      ICNT = 101
C      GENERATE THE SLAM OUTPUT NECESSARY TO DESCRIBE EACH NODE.
C      THIS WILL CONSIST OF AN AWAIT CARD, BRANCH CARDS (ACT) FOR
C      EACH MESSAGE, PLUS A SET OF PROCESS AND TRANSMIT ACTIVITIES
C      FOR EACH MESSAGE USING THIS NODE.
      DO 950 I = 1, NXTNOD
C      THREE TYPES OF AWAIT STATEMENTS ARE POSSIBLE: AN AWAIT WITHOUT
C      ANY BLOCKING OR BALKING, AN AWAIT WITH BLOCKING, OR AN AWAIT
C      WITH BALKING.
      IF (BALKS(I) .EQ. '      ') THEN
C          NO BLOCKING OR BALKING.
          WRITE (20, 40) NODNAM(I), I, NODCAP(1, I), NODNAM(I)
40          FORMAT (A4, 2X, 'AWAIT(', I4, '/', I4, ','),', A4, ';')
      ELSE IF (BALKS(I) .EQ. 'BLOK') THEN
C          BLOCKING.
          WRITE (20, 41) NODNAM(I), I, NODCAP(1, I), NODNAM(I)
41          FORMAT (A4, 2X, 'AWAIT(', I4, '/', I4, ','),', A4, ',BLOCK;')
      ELSE
C          BALKING.
          WRITE (20, 42) NODNAM(I), I, NODCAP(1, I), NODNAM(I),
&          BALKS(I)
42          FORMAT (A4, 2X, 'AWAIT(', I4, '/', I4, ','),', A4, ',BALK(',
&          A4, ');')
      ENDIF
C      PLACE A DUMMY ACTIVITY FOLLOWING THE AWAIT.
38      FORMAT ('      ACT;')
      WRITE (20,29)
C      COON TO SPECIFY ONLY ONE BRANCH TO BE TAKEN.
29      FORMAT ('      COON,1;')
C      SAVE THE NUMBER OF THE FIRST PROGRAM GENERATED LABEL.
      ISTRT = ICNT
      DO 925 J = 1, NXTMSG
C      GET THE NODE NUMBER OF THE I-TH NODE IN PATH J.
      NUM = PTHMAT (I, J)
C      IF 0, MESSAGE I DOES NOT USE PATH J.
      IF (NUM .NE. 0) THEN
C          CONSTRUCT SLAM BRANCH TO THAT PORTION OF THE SLAM CODE
C          WHICH WILL HANDLE THE PROCESS AND TRANSMIT TIMES FOR THIS
C          MESSAGE.
          WRITE (20, 500) J, ICNT
500      FORMAT ('      ACT,,ATRIB(2).EQ.',I4,',L',I3,';')
C          INCREMENT PROGRAM GENERATED LABEL NUMBER.
          ICNT = ICNT + 1
      ENDIF
925      CONTINUE
C      GENERATE A PROCESS AND TRANSMIT ACTIVITY PAIR FOR EACH MESSAGE
C      THAT USES NODE J.
      DO 926 J = 1, NXTMSG
C      GET THE NODE NUMBER OF THE I-TH NODE IN PATH J.

```

```

NUM = PTHMAT(I, J)
C   IF 0, MESSAGE I DOES NOT USE PATH J.
C   IF (NUM .NE. 0) THEN
      WRITE (20, 501) ISTRT
C     CONSTRUCT DUMMY GOON WITH PROGRAM GENERATED LABEL.
501  FORMAT ('L', I3, ' GOON;')
C     INCREMENT PROGRAM GENERATED LABEL NUMBER.
      ISTRT = ISTRT + 1
C     EACH DISTRIBUTION RATE REQUIRES A CERTAIN NUMBER OF INPUT
C     PARAMETERS.  THE FOLLOWING SELECTS THE APPROPRIATE WRITE
C     STATEMENT FOR THE DISTRIBUTION.
C
C     IF THE DISTRIBUTION IS A CONSTANT ZERO, DO NOT WRITE ANYTHING.
      IF (DSTPRC(J, I, 1) .EQ. 9.0 .AND.
          & DSTPRC(J, I, 2) .EQ. 0.0) THEN
          GO TO 600
      ENDIF
      IF (DSTPRC(J, I, 1) .EQ. 9.0) THEN
C        CONSTANT PROCESS TIME.
          WRITE (20, 502) DSTPRC(J, I, 2)
502  FORMAT (6X, 'ACT,', F10.5, ';')
      ELSE IF (DSTPRC(J, I, 1) .LE. 2.0) THEN
C        FOR EXPON AND NPSSN (REQUIRING ONE PARAMETER).
          WRITE (20, 503) DSTNAM(INT(DSTPRC(J, I, 1))),
          & DSTPRC(J, I, 2)
503  FORMAT (6X, 'ACT,', A5, '(', F10.5, ')')
      ELSE IF (DSTPRC(J, I, 1) .LE. 7.0) THEN
C        FOR BETA, GAMMA, RLOGN, RNORM AND UNFRM (REQUIRING TWO
C        PARAMETERS).
          WRITE (20, 504) DSTNAM(INT(DSTPRC(J, I, 1))),
          & DSTPRC(J, I, 2), DSTPRC(J, I, 3)
504  FORMAT (6X, 'ACT,', A5, '(', F10.5, ',', F10.5, ')')
      ELSE
C        FOR TRIAG (REQUIRING THREE PARAMETERS).
          WRITE (20, 505) DSTNAM(INT(DSTPRC(J, I, 1))),
          & DSTPRC(J, I, 2), DSTPRC(J, I, 3), DSTPRC(J, I, 4)
505  FORMAT (6X, 'ACT,', A5, '(', 2(F10.5, ','), F10.5, ')')
      ENDIF
C     PLACE A DUMMY GOON BETWEEN PROCESS AND TRANSMIT ACTIVITIES.
      WRITE (20, 506)
506  FORMAT ('          GOON;')
600  CONTINUE
C     FIND THE NODE NAME OF THE NEXT NODE IN THE PATH.  IF
C     NUM > 0, THEN THE NEXT NODE IS A NODE.  IF < 0, THEN THE
C     NEXT NODE IS A TERMINATE NODE.
      IF (NUM .GT. 0) THEN
          BNNAME = NODNAM(NUM)
      ELSE
          BNNAME = TRMNAM(-NUM)
      ENDIF
C     OBTAIN THE NUMBER OF SERVERS FOR THIS TRANSMIT ACTIVITY.

```

```

C      EACH DISTRIBUTION RATE REQUIRES A CERTAIN NUMBER OF INPUT
C      PARAMETERS.  THE FOLLOWING SELECTS THE APPROPRIATE WRITE
C      STATEMENT FOR THE DISTRIBUTION.
C      IF THE DISTRIBUTION IS A CONSTANT ZERO, DO NOT WRITE ANYTHING.
C      IF (DSTXMT(J, I, 1) .EQ. 9.0 .AND.
&        DSTXMT(J, I, 2) .EQ. 0.0) THEN
      GO TO 601
      ENDIF
      IF (DSTXMT(J, I, 1) .EQ. 9.0) THEN
      CONSTANT PROCESS TIME.
      WRITE (20, 502) DSTXMT(J, I, 2)
      ELSE IF (DSTXMT(J, I, 1) .LE. 2.0) THEN
      FOR EXPON AND NPSSN (REQUIRING ONE PARAMETER).
      WRITE (20, 503) DSTNAM(INT(DSTXMT(J, I, 1))),
      DSTXMT(J, I, 2)
      ELSE IF (DSTXMT(J, I, 1) .LE. 7.0) THEN
      FOR BETA, GAMMA, RLOGN, RNORM AND UNFRM (REQUIRING TWO
      PARAMETERS).
      WRITE (20, 504) DSTNAM(INT(DSTXMT(J, I, 1))),
      &      DSTXMT(J, I, 2), DSTXMT(J, I, 3)
      ELSE
      FOR TRIAC (REQUIRING THREE PARAMETERS).
      WRITE (20, 505) DSTNAM(INT(DSTXMT(J, I, 1))),
      &      DSTXMT(J, I, 2), DSTXMT(J, I, 3), DSTXMT(J, I, 4)
      ENDIF
      601  CONTINUE
      WRITE (20, 303) NODNAM(I)
      303  FORMAT ('      FREE,', A4, ';')
      WRITE (20, 35) BRNAME
      ENDIF
      926  CONTINUE
      950  CONTINUE
      DO 960 I = 1, NXTTRM
C      PRINT EACH OF THE TERMINATE NODE CARDS.
      WRITE (20, 50) TRMNAM(I), TRMNAM(I)
      50  FORMAT (A4, 2X, 'COLCT,INT(1),EXIT INTRVL ', A4, ';')
      WRITE (20, 55)
      55  FORMAT ('      TERM;')
      960  CONTINUE
      WRITE (20, 59)
      59  FORMAT ('      END;')
      WRITE (20, 60)
      60  FORMAT ('INIT,0;')
      WRITE (20, 65)
      65  FORMAT ('FIN;')
      RETURN
      END

```

SUBROUTINE CHKBLK

C THIS SUBROUTINE VERIFIES THAT ALL BALK PATHS ARE VALID, I.E.,
C ALL PATHS END AT A TERMINATE NODE AND NO PATH CONTAINS A CYCLE.

C CALLED BY:
C MAIN
C SUBROUTINES CALLED:
C NONE
C FUNCTIONS CALLED:
C NUMNOD
C INPUT ARGUMENTS:
C NONE
C OUTPUT ARGUMENTS:
C NONE
C INPUTS VIA COMMON:
C IFATAL
C NXTNOD
C NXTMSG
C PTHMAT
C NODNAM
C BALKS
C OUTPUTS VIA COMMON:
C IFATAL
C WORKING VARIABLES:
C I--DO LOOP COUNTER
C J--DO LOOP COUNTER
C NEXT--NEXT NODE NUMBER ON THE PATH
C NOSTEP--COUNT OF THE NUMBER OF NODES IN A PATH
C NUM--BALK NODE NUMBER

C
C PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
C CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4,
C & BALKS*4, DSTNAM*5
C COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
C & MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
C & CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
C & PARM1, PARM2, PARM3, PARM4, PARM5,
C & DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
C & LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C DO 10 I = 1, NXTNOD
C IF BALK(I) IS BLANK OR 'BLOK', THEN A BALK ROUTE WAS NOT
C SPECIFIED FOR NODE I AND, THEREFORE, NEED NOT BE CHECKED.
C IF (BALKS(I) .NE. ' ' .AND. BALKS(I) .NE. 'BLOK') THEN
C FIND THE NODE NUMBER THAT NODE I WILL BALK TO.
C NUM = NUMNOD(BALKS(I))
C DO 20 J = 1, NXTMSG
C IF PTHMAT(I,J) = 0, THEN MESSAGE J DOES NOT USE NODE I AND
C CAN BE IGNORED. OTHERWISE, FOR EACH MESSAGE J IN NODE I
C INSURE THAT BALK NODE NUM HAS A PATH FOR MESSAGE J (I.E.,
C IF MESSAGE J APPEARS IN NODE I, MESSAGE J MUST ALSO APPEAR
C IN NODE NUM).
C IF (PTHMAT(I, J) .NE. 0) THEN

```

C      NOSTEP WILL COUNT HOW MANY NODES ARE IN THE MESSAGE'S PATH.
C      NOSTEP = 0
C      FIND THE NEXT NODE NUMBER IN THE BALK PATH FOR MESSAGE J.
C      NEXT = PTHMAT(NUM, J)
30      IF (NEXT .EQ. 0) THEN
C          BALK PATH FOR MESSAGE J WAS NOT SPECIFIED OR WAS
C          INCOMPLETE (DID NOT END ON A TERMINATE NODE).
C          PRINT 50, J, NODNAME(I)
C          PRINT *, ' SUPPLY BALK PATH, OR MAKE SURE PATH ',
&          'ENDS AT A TERMINATE NODE.'
C          IFATAL = IFATAL + 1
C      ELSE
C          IF > 0, NEXT NODE IS A REGULAR NODE. IF < 0, NEXT
C          NODE IS A TERMINATE NODE (MEANING THIS PATH IS
C          SUCCESSFULLY COMPLETED).
C          IF (NEXT .GT. 0) THEN
C              FIND THE NEXT NODE NUMBER IN THE BALK PATH FOR MESSAGE
C              J.
C              NEXT = PTHMAT(NEXT, J)
C              INCREMENT THE NUMBER OF NODES IN THIS BALK PATH.
C              NOSTEP = NOSTEP + 1
C              IF MORE STEPS IN THE PATH WAS USED THAN NODES WERE
C              DEFINED, THEN A CYCLE MUST EXIST FOR THIS PATH
C              (MESSAGE J LOOPS BACK UPON ITSELF).
C              IF (NOSTEP .GT. NXTNOD) THEN
C                  PRINT 55, J, NODNAME(I)
C                  IFATAL = IFATAL + 1
C                  GO TO 20
C              ENDIF
C              GO TO 30
C          ENDIF
C      ENDIF
C      CONTINUE
20      ENDIF
10      CONTINUE
50      FORMAT (' ERROR--INCOMPLETE BALK PATH FOR MESSAGE ', I4,
& ' FROM NODE ', A4, '.')
55      FORMAT (' ERROR--THE BALK PATH FOR MESSAGE ', I4,
& ' CONTAINS A CYCLE FROM NODE ', A4, '.')
      RETURN
      END

```

```

SUBROUTINE CHKUSD

C
C THIS SUBROUTINE VERIFIES THAT EACH NODE SPECIFIED
C BY A NODE CARD IS USED IN AT LEAST ONE PATH.

C
C CALLED BY:
C     MAIN
C SUBROUTINES CALLED:
C     NONE
C FUNCTIONS CALLED:
C     NONE
C INPUT ARGUMENTS:
C     NONE
C OUTPUT ARGUMENTS:
C     NONE
C INPUTS VIA COMMON:
C     IFATAL
C     NXTNOD
C     NXTMSG
C     PTHMAT
C     NODNAM
C OUTPUTS VIA COMMON:
C     IFATAL
C WORKING VARIABLES:
C     I--DO LOOP COUNTER
C     J--DO LOOP COUNTER
C
PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4,
& BALKS*4, DSTNAM*5
COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
& MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
& CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
& PARM1, PARM2, PARM3, PARM4, PARM5,
& DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
& LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C
SEARCH EACH NODE I LOOKING FOR A MESSAGE J THAT USES NODE I
C (I.E., PTHMAT(I, J) .NE. 0).
DO 20, I = 1, NXTNOD
DO 10, J = 1, NXTMSG
IF (PTHMAT(I, J) .NE. 0.0) THEN
    THE NODE WAS USED.
    GO TO 20
ENDIF
10  CONTINUE
PRINT *, 'ERROR--NODE ', NODNAM(I), ' IS NOT USED IN A PATH.'
PRINT *, ' REMOVE CORRESPONDING NODE CARD.'
IFATAL = IFATAL + 1
20  CONTINUE
RETURN
END

```

```

SUBROUTINE MESS (LBL, PRM1, PRM2, PRM3, PRM4,
& PRM5, LBL2)
C
C THIS SUBROUTINE HANDLES MESS CARDS BY SAVING THE NECESSARY
C MESSAGE INFORMATION.
C
C CALLED BY:
C     MAIN
C SUBROUTINES CALLED:
C     NONE
C FUNCTIONS CALLED:
C     NUMDST
C     NUMNOD
C INPUT ARGUMENTS:
C     LBL--MESSAGE DISTRIBUTION NAME
C     PRM1--DISTRIBUTION PARAMETER 1
C     PRM2--DISTRIBUTION PARAMETER 2
C     PRM3--DISTRIBUTION PARAMETER 3
C     PRM4--MESSAGE START TIME
C     PRM5--NUMBER OF MESSAGE RELEASES
C     LBL2--NAME OF FIRST NODE IN PATH
C OUTPUT ARGUMENTS:
C     NONE
C INPUTS VIA COMMON:
C     IFATAL
C     NXTMSG
C     MAXMSG
C OUTPUTS VIA COMMON:
C     IFATAL
C     NXTMSG
C     CREATE
C     LASNOD
C     NODSTR
C     DIST
C WORKING VARIABLES:
C     NUMN--NODE NUMBER
C
PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
CHARACTER LBL*5, LBL2*4
CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4,
& BALKS*4, DSTNAM*5
COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
& MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
& CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
& PARM1, PARM2, PARM3, PARM4, PARM5,
& DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
& LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C
C INCREMENT MESSAGE COUNT.
NXTMSG = NXTMSG + 1
C
C INSURE NXTMSG STILL IN RANGE.
IF (NXTMSG .GT. MAXMSG) THEN
    PRINT *, 'ERROR--TOO MANY MESSAGES DEFINED.'
    IFATAL = IFATAL + 1

```

```

NXTMSG = MAXMSG
LASNOD(NXTMSG) = 1
GO TO 999
ENDIF
C   VERIFY LABL IS A VALID DISTRIBUTION NAME.
C   IF (NUMDST(LABL) .GT. 0) THEN
C     VALID NAME, SAVE IT.
C     DIST(NXTMSG) = LABL
C   ELSE
C     PRINT *, 'ERROR--ILLEGAL DISTRIBUTION SPECIFIED.'
C     IFATAL = IFATAL + 1
C   ENDIF
C   SAVE 1ST PARAMETER OF DISTRIBUTION
CREATE(1, NXTMSG) = PRM1
C   SAVE 2ND PARAMETER OF DISTRIBUTION.
CREATE(2, NXTMSG) = PRM2
C   SAVE 3RD PARAMETER OF DISTRIBUTION.
CREATE(3, NXTMSG) = PRM3
C   SAVE STARTING TIME OF THIS MESSAGE.
CREATE(4, NXTMSG) = PRM4
C   SAVE THE NUMBER OF MESSAGE RELEASES.
CREATE(5, NXTMSG) = PRM5
C   FIND THE NODE NUMBER OF THE FIRST NODE IN THIS MESSAGE.
NUMN = NUMNOD(LABL2)
IF (NUMN .GT. 0) THEN
  VALID NODE NUMBER--INITIALIZE THE LAST NODE IN THE PATH
  (LASNOD) AND THE FIRST NODE IN THE PATH (NODSTR) TO NUMN.
  LASNOD(NXTMSG) = NUMN
  NODSTR(NXTMSG) = NUMN
ELSE
  PRINT *, 'ERROR--UNDEFINED NODE.'
  PRINT *, '      CORRECT NODE NAME OR INSERT CORRESPONDING ',
&           'NODE CARD.'
  LASNOD(NXTMSG) = 1
  IFATAL = IFATAL + 1
ENDIF
999 RETURN
END

```

```

SUBROUTINE NODE (LBL, PRM1, PRM2)

C THIS SUBROUTINE HANDLES NODE CARDS BY SAVING THE NODE INFORMATION
C
C CALLED BY:
C   MAIN
C SUBROUTINES CALLED:
C   NONE
C FUNCTIONS CALLED:
C   NUMNOD
C INPUT ARGUMENTS:
C   LBL--NAME OF NODE BEING DEFINED.
C   PRM1--MAXIMUM CAPACITY OF NODE
C   PRM2--NUMBER OF SERVERS
C OUTPUT PARAMETERS:
C   NONE
C INPUTS VIA COMMON:
C   IFATAL
C   NXTNOD
C   MAXNOD
C   LABEL2
C OUTPUTS VIA COMMON:
C   IFATAL
C   NXTNOD
C   NODCAP
C   NODNAM
C   BALKS
C WORKING VARIABLES:
C   NONE

C
C   PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
C   CHARACTER LBL*5
C   CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAME*4,
C   & BALKS*4, DSTNAME*5
C   COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
C   & MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
C   & CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
C   & PARM1, PARM2, PARM3, PARM4, PARM5,
C   & DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C   COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAME(NOTRM), CRDTYP,
C   & LABEL, LABEL2, BALKS(NONOD), DSTNAME(9)
C   HAS THIS NODE BEEN PREVIOUSLY DEFINED (AS INDICATED BY A
C   VALUE > 0)?
C   IF (NUMNOD(LBL) .GT. 0) THEN
C     PRINT *, 'ERROR--NODE PREVIOUSLY DEFINED.'
C     IFATAL = IFATAL + 1
C     GO TO 999
C   ENDIF
C   INCREMENT NODE COUNT.
C   NXTNOD = NXTNOD + 1
C   VERIFY NODE COUNT STILL IN RANGE.
C   IF (NXTNOD .GT. MAXNOD) THEN
C     PRINT *, 'ERROR--TOO MANY NODES DEFINED (IGNORED).'
C     NXTNOD = MAXNOD

```

```

        IFATAL = IFATAL + 1
        NXTNOD = MAXNOD
        GO TO 999
    ENDIF
    C     SAVE NODE NAME.
    C     NODNAM(NXTNOD) = LABL
    C     VERIFY VALIDITY OF MAXIMUM CAPACITY.
    IF (PRM1 .LE. 0.0) THEN
        PRINT *, 'ERROR--MAXIMUM CAPACITY MUST BE GREATER THAN 0.'
        IFATAL = IFATAL + 1
    ELSE
        NODCAP(1, NXTNOD) = PRM1
    ENDIF
    C     NUMBER OF SERVERS MUST BE GREATER THAN 0.
    IF (PRM2 .GT. 0.0) THEN
        SAVE THE NUMBER OF SERVERS.
        NODCAP(2, NXTNOD) = PRM2
    ELSE
        PRINT *, 'ERROR--NUMBER OF SERVERS MUST BE GREATER THAN 0.'
        IFATAL = IFATAL + 1
    ENDIF
    C     IS THIS NODE BLOCKED OR NO BALK NODE SPECIFIED?
    IF (LABEL2 .EQ. 'BLOK' .OR. LABEL2 .EQ. ' ') THEN
        ITS BLOCKED OR NO BALK NODE SPECIFIED.
        BALKS(NXTNOD) = LABEL2
    ELSE
        IS THE BALK NODE NAME VALID?
        IF (NUMNOD(LABEL2) .GT. 0) THEN
            ITS VALID--SAVE IT.
            BALKS(NXTNOD) = LABEL2
        ELSE
            C     ERROR--ITS NOT VALID.
            PRINT *, 'ERROR--ILLEGAL BALK NODE.'
            PRINT *, '          CORRECT THE NAME OR INSERT ',
&           'CORRESPONDING NODE CARD.'
            IFATAL = IFATAL + 1
        ENDIF
    ENDIF
999  RETURN
END

```

SUBROUTINE PATH

C THIS SUBROUTINE HANDLES PATH CARDS BY SAVING THE INFORMATION
C NECESSARY TO CONSTRUCT A PATH.

C CALLED BY:
C MAIN

C SUBROUTINES CALLED:
C NONE

C FUNCTIONS CALLED:
C NUMNOD
C NUMTRM

C INPUT ARGUMENTS:
C NONE

C OUTPUT ARGUMENTS:
C NONE

C INPUTS VIA COMMON:
C IFATAL
C NXTMSG
C PTHMAT
C LASNOD
C LABEL

C OUTPUTS VIA COMMON:
C IFATAL
C PTHMAT
C LASNOD

C WORKING VARIABLES:
C NUMN--NODE NUMBER
C NUMT--TERMINATE NODE NUMBER

C
C PARAMETER (NONOD=10, NOTRM=10, NOMSG=10)
C CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4
& BALKS*4, DSTNAM*5
C COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
& MAXMSC, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
& CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
& PARM1, PARM2, PARM3, PARM4, PARM5,
& DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
& LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)

C FIND NODE NUMBER FOR THIS LABEL.
C NUMN = NUMNOD(LABEL)

C FIND TERMINATE NODE NUMBER FOR THIS LABEL.
C NUMT = NUMTRM(LABEL)

C IF BOTH NUMN AND NUMT ARE 0, THEN THIS LABEL IS NEITHER A
C TERMINATE NOR A REGULAR NODE.

IF (NUMN .EQ. 0 .AND. NUMT .EQ. 0) THEN
PRINT *, 'ERROR--UNDEFINED NODE.'
PRINT *, ' CORRECT NODE NAME OR INSERT CORRESPONDING ',
& 'NODE OR TERM CARD.'
IFATAL = IFATAL + 1

ELSE
C VERIFY A PATH HAS NOT BEEN PREVIOUSLY DEFINED AT THIS LOCATION
IF (PTHMAT(LASNOD(NXTMSG), NXTMSG) .NE. 0.0) THEN

```
      PRINT *, 'ERROR--DUPLICATE PATH DEFINITION.'.
      PRINT *, '          (A PATH FOR THIS MESSAGE AT THIS NODE WAS
&          , 'PREVIOUSLY SPECIFIED.)'
      IFATAL = IFATAL + 1
      ELSE
      IF (NUMN .GT. 0) THEN
C          A REGULAR NODE--MOVE THIS NODE NUMBER INTO THE PATH
C          MATRIX (PTHMAT).
C          PTHMAT(LASNOD(NXTMSG), NXTMSG) = NUMN
C          THIS NODE NOW BECOMES THE PREVIOUS (LAST) NODE IN THIS
C          PATH.
C          LASNOD(NXTMSG) = NUMN
      ELSE
C          A TERMINATE NODE--MOVE THE NEGATIVE OF THE TERMINATE NODE
C          NUMBER INTO THE PATH MATRIX (PTHMAT).
C          PTHMAT(LASNOD(NXTMSG), NXTMSG) = -NUMT
C          THE TERMINATE NODE NOW BECOMES THE PREVIOUS (LAST) NODE
C          IN THIS PATH.
C          LASNOD(NXTMSG) = -NUMT
      ENDIF
      ENDIF
      RETURN
END
```

```

SUBROUTINE PROC(LBL, PRM1, PRM2, PRM3)
C
C THIS SUBROUTINE HANDLES PROC CARDS BY SAVING THE INFORMATION
C NECESSARY FOR PROCESSING TIMES.
C
C CALLED BY:
C     MAIN
C SUBROUTINES CALLED:
C     NONE
C FUNCTIONS CALLED:
C     NUMDST
C INPUT ARGUMENTS:
C     LBL--PROCESSING TIME DISTRIBUTION NAME
C     PRM1--1ST PARAMETER OF DISTRIBUTION
C     PRM2--2ND PARAMETER OF DISTRIBUTION
C     PRM3--3RD PARAMETER OF DISTRIBUTION
C OUTPUT ARGUMENTS:
C     NONE
C INPUTS VIA COMMON:
C     IFATAL
C     NXTMSG
C     LASNOD
C OUTPUTS VIA COMMON:
C     IFATAL
C     DSTPRC
C WORKING VARIABLES:
C     NUM--DISTRIBUTION RATE NUMBER
C
PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
CHARACTER LBL*5
CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4,
& BALKS*4, DSTNAM*5
COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
& MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
& CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
& PARM1, PARM2, PARM3, PARM4, PARM5,
& DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
& LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C
FIND THE DISTRIBUTION NUMBER.
NUM = NUMDST(LBL)
C
IS DISTRIBUTION VALID (I.E., > 0)?
IF (NUM .EQ. 0) THEN
    PRINT *, 'ERROR--ILLEGAL DISTRIBUTION SPECIFIED.'
    IFATAL = IFATAL + 1
ELSE
    VALID, SAVE DISTRIBUTION NUMBER.
    DSTPRC(NXTMSG, LASNOD(NXTMSG), 1) = NUM
    SAVE DISTRIBUTION PARAMETER 1.
    DSTPRC(NXTMSG, LASNOD(NXTMSG), 2) = PRM1
    SAVE DISTRIBUTION PARAMETER 2.
    DSTPRC(NXTMSG, LASNOD(NXTMSG), 3) = PRM2
    SAVE DISTRIBUTION PARAMETER 3.
    DSTPRC(NXTMSG, LASNOD(NXTMSG), 4) = PRM3

```

ENDIF
RETURN
END

```

SUBROUTINE READ
C
C      THIS SUBROUTINE READS THE INPUT DATA DECK FROM FILE 20.
C
C      CALLED BY:
C          MAIN
C      SUBROUTINES CALLED:
C          NONE
C      FUNCTIONS CALLED:
C          NONE
C      INPUT ARGUMENTS:
C          NONE
C      OUTPUT ARGUMENTS:
C          NONE
C      INPUTS VIA COMMON:
C          IFATAL
C      OUTPUTS VIA COMMON:
C          IFATAL
C          PARM1
C          PARM2
C          PARM3
C          PARM4
C          PARM5
C          CRDTYP
C          LABEL
C          LABEL2
C      WORKING VARIABLES:
C          CARD67--CARD COLUMNS 6-72 OF THE INPUT CARD
C
PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
CHARACTER CARD67*67
CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4,
& BALKS*4, DSTNAM*5
COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
& MAXMSG, NODCAP(2, NONOD), PTIMAT(NONOD, NOMSG),
& CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
& PARM1, PARM2, PARM3, PARM4, PARM5,
& DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
& LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C      READ A CARD.
READ (10, 10, END=999) CRDTYP, CARD67
PRINT 30, CRDTYP, CARD67
C      RE-READ EACH CARD BASED UPON THE CARD TYPE TO OBTAIN THE NECESSARY
C      INFORMATION FOR THAT PARTICULAR CARD TYPE.
IF (CRDTYP .EQ. 'NODE') THEN
    BACKSPACE 10
    READ (10, 20, ERR=888) LABEL, PARM2, PARM3,
& LABEL2
ELSE IF (CRDTYP .EQ. 'TERM' .OR. CRDTYP .EQ. 'PATH') THEN
    BACKSPACE 10
    READ (10, 40, ERR=888) LABEL
ELSE IF (CRDTYP .EQ. 'MESS') THEN
    BACKSPACE 10

```

```
      READ (10, 50, ERR=888) LABEL, PARM1, PARM2, PARM3,
&      PARM4, PARM5, LABEL2
      ELSE IF (CRDTYP .EQ. 'PROC' .OR. CRDTYP .EQ. 'XMIT') THEN
          BACKSPACE 10
          READ (10, 70, ERR=888) LABEL, PARM3, PARM4, PARM1
      ELSE IF (CRDTYP .EQ. 'BALK') THEN
          BACKSPACE 10
          READ (10, 90, ERR=888) LABEL, PARM1
      ELSE IF (CRDTYP .NE. 'END ') THEN
          C      AN INVALID CARD TYPE WAS SPECIFIED.
          PRINT 100
          IFATAL = IFATAL + 1
      ENDIF
      GO TO 1000
888  PRINT 110
      CRDTYP = 'ERR '
      IFATAL = IFATAL + 1
1000 RETURN
999  PRINT 120
      STOP
10  FORMAT (A4, A67)
20  FORMAT (5X, A5, 2F10.5, A4)
30  FORMAT (1X,A4, A67)
40  FORMAT (5X, A5)
50  FORMAT (5X, A5, 5F10.5, A4)
70  FORMAT (5X, A5, 3F10.5)
90  FORMAT (5X, A5, F10.5)
100 FORMAT (' ERROR--INVALID CARD TYPE.')
110 FORMAT (' ERROR--CONVERSION ERRORS OCCURRED.', /,
&      7X, 'CHECK INPUT FORMATS.')
120 FORMAT (' UNEXPECTED END-OF-FILE.  JOB TERMINATED.')
END
```

```

SUBROUTINE TERM (LABL)
C
C THIS SUBROUTINE HANDLES TERM CARDS BY SAVING THE TERMINATE NODE
C NAME.

C
C CALLED BY:
C     MAIN
C
C SUBROUTINES CALLED:
C     NONE
C
C FUNCTIONS CALLED:
C     NUMTRM
C
C INPUT ARGUMENTS:
C     LABL--TERMINATE NODE NAME
C
C OUTPUT ARGUMENTS:
C     NONE
C
C INPUTS VIA COMMON:
C     IFATAL
C     NXTTRM
C     MAXTRM
C
C OUTPUTS VIA COMMON:
C     IFATAL
C     NXTTRM
C     TRMNAM
C
C WORKING VARIABLES:
C     NONE
C
C
C PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
C CHARACTER LABL*5
C CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4
& BALKS*4, DSTNAM*5
C COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
& MAXMSG, NODCAP(2, NONOD), PTIMAT(NONOD, NOMSG),
& CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
& PARM1, PARM2, PARM3, PARM4, PARM5,
& DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
& LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C
C HAS THIS TERMINATE NODE BEEN PREVIOUSLY DEFINED?
C
IF (NUMTRM(LABL) .GT. 0) THEN
    PRINT *, 'ERROR--TERM PREVIOUSLY DEFINED.'
    IFATAL = IFATAL + 1
ELSE
    INCREMENT TERMINATE NODE COUNT.
    NXTTRM = NXTTRM + 1
C
C IS COUNT STILL IN RANGE?
C
IF (NXTTRM .GT. MAXTRM) THEN
    PRINT *, 'ERROR--TOO MANY TERMS DEFINED (IGNORED).'
    IFATAL = IFATAL + 1
    NXTTRM = MAXTRM
ELSE
    SAVE THIS TERMINATE NODE NAME.
    TRMNAM(NXTTRM) = LABL
ENDIF
ENDIF

```

RETURN
END

```

C      SUBROUTINE XMIT (LBL, PRM1, PRM2, PRM3)
C
C      THIS SUBROUTINE HANDLES XMIT CARDS BY SAVING THE INFORMATION
C      NECESSARY FOR TRANSMIT TIME.
C
C      CALLED BY:
C          MAIN
C      SUBROUTINES CALLED:
C          NONE
C      FUNCTIONS CALLED:
C          NUMDST
C      INPUT ARGUMENTS:
C          LBL--TRANSMIT TIME DISTRIBUTION NAME
C          PRM1--1ST PARAMETER OF DISTRIBUTION
C          PRM2--2ND PARAMETER OF DISTRIBUTION
C          PRM3--3RD PARAMETER OF DISTRIBUTION
C      OUTPUT ARGUMENTS:
C          NONE
C      INPUTS VIA COMMON:
C          IFATAL
C          NXTMSG
C          LASNOD
C      OUTPUTS VIA COMMON:
C          IFATAL
C          DSTXMT
C      WORKING VARIABLES:
C          NUM--DISTRIBUTION RATE NUMBER
C
C      PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
C      CHARACTER LBL*5
C      CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4.
C      & BALKS*4, DSTNAM*5
C      COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
C      & MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
C      & CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
C      & PARM1, PARM2, PARM3, PARM4, PARM5,
C      & DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C      COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
C      & LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C
C      FIND THE DISTRIBUTION NUMBER.
C      NUM = NUMDST(LBL)
C
C      IS DISTRIBUTION NUMBER VALID?
C      IF (NUM .EQ. 0) THEN
C          PRINT *, 'ERROR--ILLEGAL DISTRIBUTION SPECIFIED.'
C          IFATAL = IFATAL + 1
C      ELSE
C          SAVE THE DISTRIBUTION NUMBER.
C          DSTXMT(NXTMSG, LASNOD(NXTMSG), 1) = NUM
C          SAVE PARAMETER 1 OF THE DISTRIBUTION.
C          DSTXMT(NXTMSG, LASNOD(NXTMSG), 2) = PRM1
C          SAVE PARAMETER 2 OF THE DISTRIBUTION.
C          DSTXMT(NXTMSG, LASNOD(NXTMSG), 3) = PRM2
C          SAVE PARAMETER 3 OF THE DISTRIBUTION.
C          DSTXMT(NXTMSG, LASNOD(NXTMSG), 4) = PRM3

```

ENDIF
RETURN
END

```

FUNCTION NUMDST(NAME)
C
C      THIS FUNCTION RETURNS THE DISTRIBUTION NUMBER FOR NAME (IF
C      A VALID DISTRIBUTION NAME) OR 0 (IF INVALID).
C
C      CALLED BY:
C          BLDOUT
C          MESS
C          PROC
C          X'IT
C      SUBROUTINES CALLED:
C          NONE
C      FUNCTIONS CALLED:
C          NONE
C      INPUT ARGUMENTS:
C          NAME--DISTRIBUTION NAME
C      OUTPUT ARGUMENTS:
C          NONE
C      INPUTS VIA COMMON:
C          DSTNAM
C      OUTPUTS VIA COMMON:
C          NONE
C      WORKING VARIABLES:
C          I--DO LOOP COUNTER
C
C      PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
C      CHARACTER NAME*5
C      CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4
&      BALKS*4, DSTNAM*5
C      COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
&      MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
&      CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
&      PARM1, PARM2, PARM3, PARM4, PARM5,
&      DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C      COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
&      LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C      ESTABLISH DEFAULT RETURN VALUE.
NUMDST = 0
DO 10 I = 1, 9
C      IF FOUND, RETURN I.
IF (NAME .EQ. DSTNAM(I)) THEN
    NUMDST = I
    GO TO 20
ENDIF
10  CONTINUE
20  RETURN
END

```

```

FUNCTION NUMNOD(NODE)
C
C      THIS FUNCTION RETURNS THE NODE NUMBER FOR NODE (IF THE NODE HAS
C      BEEN PREVIOUSLY DEFINED) OR 0 (IF UNDEFINED).
C
C      CALLED BY:
C          MAIN
C          CHKELK
C          BALK
C          MESS
C          NODE
C          PATH
C      SUBROUTINES CALLED:
C          NONE
C      FUNCTIONS CALLED:
C          NONE
C      INPUT ARGUMENTS:
C          NODE--NODE NAME
C      OUTPUT ARGUMENTS:
C          NONE
C      INPUTS VIA COMMON:
C          MAXNOD
C          NODNAM
C      OUTPUTS VIA COMMON:
C          NONE
C      WORKING VARIABLES:
C          I--DO LOOP COUNTER
C
C      PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
C      CHARACTER NODE*4
C      CHARACTER CRDTYP*4, LABEL*4, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*4
&      BALKS*4, DSTNAM*5
C      COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
&      MAXMSG, NODCAP(2, NONOD), PTHMAT(NONOD, NOMSG),
&      CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
&      PARM1, PARM2, PARM3, PARM4, PARM5,
&      DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C      COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
&      LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C      ESTABLISH DEFAULT RETURN VALUE.
NUMNOD = 0
DO 10, I = 1, MAXNOD
C      IF FOUND, RETURN I.
IF (NODE .EQ. NODNAM(I)) THEN
    NUMNOD = I
    GO TO 20
ENDIF
10  CONTINUE
20  RETURN
END

```

```

FUNCTION NUMTRM (NODE)
C
C THIS FUNCTION RETURNS THE TERMINATE NODE NUMBER FOR NODE (IF
C THE TERMINATE NODE HAS BEEN DEFINED) OR 0 (IF UNDEFINED).
C
C CALLED BY:
C     MAIN
C     PATH
C     TERM
C
C SUBROUTINES CALLED:
C     NONE
C
C FUNCTIONS CALLED:
C     NONE
C
C INPUT ARGUMENTS:
C     NODE--TERMINATE NODE NAME
C
C OUTPUT ARGUMENTS:
C     NONE
C
C INPUTS VIA COMMON:
C     MAXTRM
C     TRMNAM
C
C OUTPUTS VIA COMMON:
C     NONE
C
C WORKING VARIABLES:
C     I--DO LOOP COUNTER
C
C
C PARAMETER (NONOD=10,NOTRM=10,NOMSG=10)
C CHARACTER NODE*4
C CHARACTER CRDTYP*4, LABEL*5, LABEL2*4, NODNAM*4, DIST*5, TRMNAM*/
C & BALKS*4, DSTNAM*5
C COMMON /COM1/ IFATAL, NXTNOD, NXTTRM, NXTMSG, MAXNOD, MAXTRM,
C & MAXMSG, NODCAP(2, NONOD), PTIMAT(NONOD, NOMSG),
C & CREATE(6, NOMSG), LASNOD(NOMSG), NODSTR(NOMSG),
C & PARM1, PARM2, PARM3, PARM4, PARM5,
C & DSTPRC(NOMSG, NONOD, 4), DSTXMT(NOMSG, NONOD, 4)
C COMMON /COM2/ NODNAM(NONOD), DIST(NOMSG), TRMNAM(NOTRM), CRDTYP,
C & LABEL, LABEL2, BALKS(NONOD), DSTNAM(9)
C
C ESTABLISH DEFAULT RETURN VALUE.
C     NUMTRM = 0
C     DO 10 I = 1, MAXTRM
C
C     IF FOUND, RETURN I.
C     IF (NODE .EQ. TRMNAM(I)) THEN
C         NUMTRM = I
C         GO TO 20
C
C     ENDIF
C     CONTINUE
C 10     RETURN
C
C END

```

Appendix B: Operating Instructions

Introduction

The SLAM Pre-Processor is designed to operate in a batch mode, with card or card image inputs from file 10. Eight legal card types describe the network to be simulated, and the flow of messages through the network.

Card Types

The eight legal card types are BALK, END, MESS, NODE, PATH, PROC, TERM, and XMIT. Each of these are described in more detail below. Samples of each card type may be found in Appendix E.

BALK Card. When the capacity of a node has been exceeded, a balk node may be specified (more details concerning balking may be found in the description of the NODE card). A BALK card is used to define the beginning of a balk route specification (the route to be taken should a node's capacity be exceeded) by specifying the name of the node for which a balk route is being provided, and the message number for which the route applies.

END Card. An END card marks the end of the SLAMPP input deck.

MESS Card. A MESS card marks the beginning of a message description and defines the message arrival rate, the time of the first message arrival, the number of message releases (number of messages to be created), and the name of the first node on the message path.

NODE Card. A NODE card defines a node (by name), its maximum capacity, the number of servers (the number of messages which may be

processed at the same time) at this node, and balking or blocking information.

When a message arrives at a node whose queue is at its maximum capacity, one of three actions may occur: the message may balk, the message may block, or the message may be lost. If balking is specified, the message will go to an alternate (balk) node. If blocking is specified, the message "backs up" in the network prior to the node and waits until a message leaves the node, at which time it moves into the node. If neither balking nor blocking is specified, the message will simply be lost.

PATH Card. A PATH card specifies the name of the node which is next in the path.

PROC Card. A PROC card describes the amount of time a message takes to be processed prior to leaving a node.

TERM Card. A TERM card defines (by name) those nodes at which a message may leave the network, called "terminate nodes."

XMIT Card. An XMIT card describes the amount of time it takes to transmit a message to the next node on the path after a message has been processed.

Distribution Rates

Three card types (MESS, PROC, and XMIT) allow the user to specify a time duration. In the case of the MESS card, the time is the time between message creations while, for PROC and XMIT, it is the time for processing and transmitting a message, respectively.

SLAMPP will recognize nine distributions as valid. Each

distribution requires from one to three parameters, which should be input in the order given. These are described below (Pritsker, 1979: 528-534).

BETA. Beta distribution with two parameters: Theta and phi.

CONST. Constant duration with one parameter: length.

EXPON. Exponential distribution with one parameter: mean.

GAMA. Gamma distribution with two parameters: Beta and alpha.

NPSSN. Poisson distribution with one parameter: mean.

RLOGN. Lognormal distribution with two parameters: mean and standard deviation.

RNORM. Normal distribution with two parameters: mean and standard deviation.

TRIAG. Triagonal distribution with three parameters: interval low value, mode, and interval high value.

UNFRM. Uniform distribution with two parameters: interval low value and interval high value.

Input Formats

Each input card consists of a card type, plus (depending upon card type) up to seven additional data areas. Regardless of card type, each data area (or field) is located in the same card columns. Field one is in columns 1-4 and field two in 6-10, while the remaining fields begin in columns 11, 21, 31, 41, 51, and 61, and are each 10 columns wide.

Two types of data may be placed within each field: alphanumeric or numeric. Alphanumeric data must be left justified within the field. All numeric data is input as real values, and may fall any place within

a field; however, a decimal point must be explicitly included with the number. Thus, the number "1" should be entered as "1." or "1.0". Only the first five decimal places are significant.

Although each numeric field should be entered as a real value (to keep all data field formats consistent for all card types), some values are internally converted to integers by truncation. Therefore, if the data item is listed as "integer," ensure that no decimal places are included.

The format for each of the eight card types is shown below. The number within parenthesis is the field number associated with that data item.

BALK. (1) Card type key word "BALK". (2) Name of the node (not a terminate node) for which a balk route is being specified. This node name should have appeared as a balk node on a NODE card. (3) The integer message number for which a balk route is being specified. Message numbers are assigned sequentially (beginning with one) in the order found in the SLAMPP input deck (see the section on message numbering later).

END. (1) Card type keyword "END".

MESS. (1) Card type keyword "MESS". (2) Distribution name for the message creation rate. (3) Parameter one of the distribution. (4) Parameter two of the distribution (if only one parameter is required, enter zero). (5) Parameter three of the distribution (if only one or two parameters are required, enter zero). (6) The integer time of creation of the first message. (7) The integer number of messages to be created. (8) The name of the first node in this message's path.

NODE. (1) Card type keyword "NODE". (2) The name of the node being defined (four or fewer characters). (3) The integer maximum capacity of this node (the maximum number of messages which may be waiting at this node at any one time). (4) The integer number of servers at this node (the maximum number of messages which may be processed concurrently). (5) Balking or blocking information. If balking is to occur, place the name of the node to which messages should balk if maximum capacity is exceeded. If blocking is to occur, place the keyword "BLOK" into this field. If neither balking or blocking is desired (meaning that an arriving message will be lost if maximum capacity has been reached), leave this field blank.

PATH. (1) Card type keyword "PATH". (2) The name of the next node along this message or balk route's path.

PROC. (1) Card type keyword "PROC". (2) The distribution name for the processing time at this node. (3) Parameter one of the distribution. (4) Parameter two of the distribution (if only one parameter is required, enter zero). (5) Parameter three of the distribution (if only one or two parameters are required, enter zero).

TERM. (1) Card type keyword "TERM". (2) The name (four or fewer characters) of the terminated node being declared.

XMIT. (1) Card type keyword "XMIT". (2) The distribution name for the transmission time at this node. (3) Parameter one of the distribution. (4) Parameter two of the distribution (if only one parameter is required, enter zero). (5) Parameter three of the distribution (if only one or two parameters are required, enter zero).

Building the Input Deck

The input deck can be divided into four categories, which are described below. The order of these descriptions is the same as the order in which these categories must fall within the SLAMPP input deck.

Declarations. Before any message can be defined, all nodes within the network must be defined. Each node must be described and defined (using NODE cards). Following the node declarations comes the declaration of all terminate nodes (using TERM cards).

Message Descriptions. Following the declaration portion of the SLAMPP input is the message description area. A message description may involve up to four card types: MESS, PROC, XMIT and PATH.

The first card in the description of any message is a MESS card specifying the creation rate for this message and the first node along the path. In essence, this card specifies when and how often this message will arrive, and the name of the node where the message will first arrive.

While at a node, a message may be processed in some manner which requires time. This processing time can be expressed by using a PROC card. If processing is not required, this card may be omitted.

After all processing is complete, this message is ready to be transmitted to the next node along the path. This transmission may require time, and can be defined using an XMIT card. If no transmission time is needed, this card may be omitted.

The next node along this message's path (that is, the node to which the message is being transmitted) is defined by a PATH card. This PROC, PATH and XMIT pattern is repeated until the entire path of the message

has been described. The last node of every message's path must be a terminate node.

Subsequent messages are described by inserting another MESS card and following the above instructions again.

Balk Route Descriptions. One of the items specified on a NODE card is whether balking will occur or not. If balking is specified, there is a possibility that every message that uses this node could be redirected to the balk node. Therefore, a path must exist for every message which could possibly enter a balk node. If a path already exists for a message at the balk node from the message's original description, nothing more need be done for this message. However, if a path does not exist, one must be supplied by means of a BALK card. All balk route descriptions (if any are present) follow the last message description.

The BALK card specifies the balk node name and the message number for which a balk route is being provided. Following the BALK card, PROC, XMIT and PATH cards are repeated (just as in a message description) until the entire balk path is defined. As with a regular message path, balk paths must end at a terminate node.

Additional balk routes may be supplied by inserting another BALK card and repeating the above steps.

End of Deck. The last card of the input deck must be an END card. No additional data will be read beyond the END card.

Design Considerations

Designing the SLAMPP Network. Experience has shown that even the most simple network can become confusing unless a systematic method is

used to translate that network into SLAMPP inputs. In order to minimize the number of input errors, a few simple guidelines are suggested.

First, determine exactly which nodes and terminate nodes will be required for the network. In the case of nodes, also determine the number of servers, maximum capacity, and balking or blocking characteristics. Translate this information into the appropriate NODE or TERM cards.

Second, determine exactly which messages are to be included. A good method to follow is to "outline" the message directly into SLAMPP inputs. That is, code the appropriate MESS card for the message, and follow this with the applicable PROC, XMIT and PATH cards to describe the message. After the first message is completely defined, begin defining the second, then the third, etc., until all messages have been defined.

Third, define any balk routes which may be required. To do this, examine each node, in succession, which has a balk node specified on its NODE card. Then, for every message which uses this node, ensure a route exists for this message from the balk node. This route may already exist if the message originally was routed using this balk node as part of its path. For example, as might occur if a message went from N1 to N2 to N3 to N4, and N1 balked to N3. In this case, if N1 was at maximum capacity the message would skip N1 and N2 and go directly to N3. If a route does not already exist for this message, then a route must be defined by using a BALK card. As with a message, follow the BALK card with the necessary PROC, XMIT and PATH cards to describe the route.

Fourth, make sure that any routes added in step three that use a

node which can balk has its own balk routes specified. Repeat this step until all routes and all necessary balk routes are specified.

Lastly, supply an END card to the input deck.

Message Numbering. Balk routes are specified using the number of the message for which a route is being supplied. It is, therefore, important to know how messages are numbered.

Messages are numbered consecutively (beginning at one) in the order they appear in the input deck. Thus, the first message found is message number one; the second, number two, etc.

When additional messages are being added to the network, always add the new message(s) after the end of the last message of the network. In this manner, any balk routes which may have been previously defined will not need any changes. Otherwise, the BALK cards will need changing to reflect the new message numbers.

Balk Route Considerations. Balk route design provides the user with several methods which may be used. To familiarize the user with these methods, they are described below.

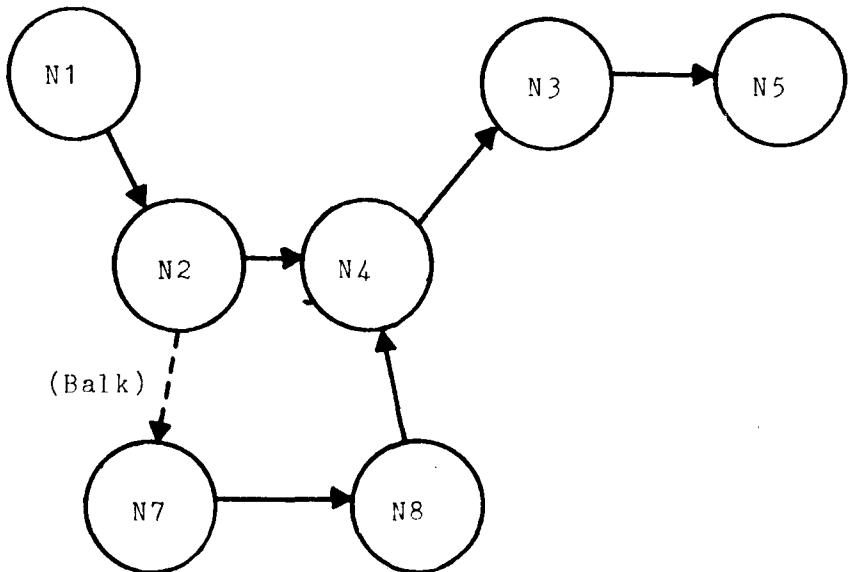
First, no balk route may be necessary and therefore, the user need not do anything. However, this is only applicable to nodes which do not have a balk node specified.

Second, the balk route may be an entirely different route from the path defined for the message (by the MESS card). The primary consideration here is to ensure that this balk route ends on a terminate node.

Third, the balk node may have been included within the message's original path. In this case, a route already exists for this message

from the balk node, and the user need not (in fact, cannot) provide another route.

Fourth, the balk route may merge back into the message route. This means that a portion of the balk route is entirely different from the message path; however, at some point in the balk route a node defined in the message path is used. From this node forward, the balk route is the same as the message path. To illustrate this, assume a message has path N1, N2, N4, N3 and N5 (in that order). If N2 balks to N7, then a merging balk route might be N7, N8 and N4. In this example, a message would balk from N2 to N7 and then traverse N8 and N4. The remainder of the path from N4 would be the same as the message path. In essence, the balk route would be N7, N8, N4, N3 and N5. Diagrammatically, this could be viewed as:



All messages which arrive at a node are required to balk to the same balk node. Occasionally, however, different balk nodes may be desired for each different message. This can be accomplished by using a "dummy" balk node whose only function is to redirect messages to their appropriate nodes. The recommended procedure is to define this dummy balk node as having a very large capacity (9999) and a large number of servers (9999). Since this node does not represent a real node within the network, following this definition prevents this node from becoming a "bottleneck" in the system and affecting statistics. Immediately following the BALK cards for the messages to be redirected, insert a PATH card to the desired node. This causes the message to balk to the dummy node and then be transmitted to the appropriate node. However, since both balking and transmitting (in this case) use zero time, the net result is that the message goes directly to its appropriate node.

Appendix C: Input Errors

The validity of the SLAM Pre-Processor input is verified at several points. Should any errors be detected, an error message is printed and the SLAM input source code is not created. Below is a list of the error messages which may be produced, along with an explanation and a possible cause or correction.

CONVERSION ERRORS OCCURRED. A conversion error occurred trying to read the input file. The most likely cause is failure to follow the correct input format for the card type involved. Correct the format.

DUPLICATE PATH DEFINITION. An attempt was made to define a path for a particular message using a node already defined on that message's path. Frequently caused by trying to define a balk path for a message which overlaps the message's path.

END CARD EXPECTED. Additional input data followed the end of the last balk route or, if no balk routes are specified, the end of the last message. Either this data is incorrect (and should be removed), or a problem exists in the last message or balk route to be specified.

INCOMPLETE BALK PATH FOR MESSAGE n FROM NODE name. The balk path for message number n (which balks from node name) does not end on a terminate node. Either the path is incorrect (and should be corrected) or the balk route was not supplied as required.

ILLEGAL BALK NODE. The balk node name specified was not defined in a NODE statement. Either supply a corresponding NODE statement, or correct the node name.

ILLEGAL BALK NODE NAME. The balk node name specified on the BALK

card is incorrect and should be corrected.

ILLEGAL DISTRIBUTION SPECIFIED. The distribution name specified is not valid in the SLANPP (see Appendix B). Correct to a valid name.

INVALID BALK NODE (SKIPPING TO NEXT BALK). The balk node name is incorrect and should be corrected. The remainder of this balk path is ignored.

INVALID CARD TYPE. The card type specified is not a valid SLAMPP card type (see Appendix B). Correct the card type or remove from the input deck.

INVALID MESSAGE NUMBER (SKIPPING TO NEXT BALK). The message number specified on the BALK card is invalid (either zero, negative, or greater than the number of messages input to SLAMPP). Correct the number. The remainder of this balk path is ignored.

MAXIMUM CAPACITY MUST BE GREATER THAN 0. The maximum capacity specified for this node was either zero or negative. Correct to a positive value.

MESS CARD MUST FOLLOW TERM CARDS. A card type other than MESS was found following the last TERM card. The input deck is out of order (and should be reordered).

MESSAGE MUST END ON A TERMINATE NODE. The message path being defined did not end on a terminate node, leaving this path incomplete. Complete the path.

MISSING BALK PATH CARD. SKIPPING TO NEXT BALK. The path for this balk route did not contain a PATH card. The remainder of this path is ignored.

MISSING PATH CARD. SKIPPING TO NEXT MESS. The path for this

message route did not contain a PATH card. The remainder of this message is ignored.

NODE CARDS MUST BE FIRST. The first card type found was not a NODE card. Inputs are out of order and should be reordered.

NODE name IS NOT USED IN A PATH. The node name, which was defined in a NODE card, was not used in any path. Either place name in a path, or remove the corresponding NODE card.

NODE PREVIOUSLY DEFINED. The node being defined by this NODE card has appeared previously on a NODE card. Remove one of the NODE cards.

NUMBER OF SERVERS MUST BE GREATER THAN 0. The number of servers specified for this node was either zero or negative. Correct to a positive value.

TERM CARDS MUST FOLLOW NODE CARDS. The first card type found following the last NODE card was not a TERM card. Inputs are out of order and should be reordered.

TERM PREVIOUSLY DEFINED. The terminate node being defined by this TERM card has appeared previously on a TERM card. Remove one of the TERM cards.

THE BALK PATH FOR MESSAGE n CONTAINS A CYCLE FROM NODE name. The balk route specified for message number n from node name loops back upon itself. Correct the path to remove the cycle.

TOO MANY MESSAGES DEFINED. The number of messages input to the SLAMPP exceeds the program capacity. Either remove the excess messages, or recompile the Pre-Processor with an increased value for NOMSG (see Chapter III).

TOO MANY NODES DEFINED (IGNORED). The number of nodes input to the

SLAMPP exceeds the program capacity. This node declaration is ignored. Either remove the excess nodes, or recompile the Pre-Processor with an increased value for NONOD (see Chapter III).

TOO MANY TERMS DEFINED (IGNORED). The number of terminate nodes input to the SLAMPP exceeds the program capacity. This terminate node declaration is ignored. Either remove the excess terminate nodes, or recompile the Pre-Processor with an increased value for NOTRM (see Chapter III).

UNDEFINED NODE. The node specified was not declared in a NODE or TERM statement. Insert the corresponding declaration.

UNEXPECTED END-OF-FILE. JOB TERMINATED. The end of the input data was encountered when SLAMPP expected additional data, usually caused by a missing END card or previous errors. Insert an END card or correct the other errors.

Appendix D: Interpreting the SLAM Output

Introduction

Upon successful completion of a network simulation, the user must be able to interpret the SLAM output. Summarized here is a description of the outputs. Refer to the sample SLAM output at the end of this appendix for examples.

SLAM Output

The output of interest to the user begins on the page labelled "SLAM SUMMARY REPORT." The first item to notice is the "Current Time." The time shown is the time the last message left the network, leaving the network empty.

Terminate Nodes

Following this is the "Statistics for Variables Based on Observations." Statistics are produced for each terminate node within the network and are identified by terminate node name. The statistics are based upon the time interval between message arrivals at the terminate node. The mean, standard deviation, and coefficient of variation of the time between message arrivals are listed. Also, the minimum and maximum arrival intervals and the number of messages to arrive at this terminate node are shown.

Queues

File statistics are printed next. These statistics deal with the

queues for each of the nodes in the network. The queue statistics are listed in the same order as their NODE cards in the SLAMPP network. An easier way, perhaps, to identify the node's name is to look at the next section of the SLAM output titled "Resource Statistics." The "Resource Number" for a particular node name (listed under "Resource Label") is the same as that node's "File Number." Note that the last file number listed is a SLAM work file and should be ignored.

The statistics shown include the average length, standard deviation and maximum length of each queue. The average time a message waited to be processed at a queue is also presented. Note that messages may have been lost at nodes which do not have balking nor blocking specified if the maximum length of that node's queue is the same as that node's maximum capacity.

Service and Transmission

The "Resource Statistics" represents statistics dealing with the service and transmission time at each node. The first group of data represents node utilization while the second group represents node availability.

In the utilization group, "Current Capacity" is the number of servers available at this node. The "Average Utilization" is the average number of servers to be used. The maximum number of servers used concurrently and the current number of servers being used (at end of simulation) are also shown.

In the available group, the average number of servers available, plus the minimum and maximum number available are shown.

S L A M S U M M A R Y R E P O R T

S I M U L A T I O N P R O J E C T S L A M

B Y S L A M P P

D A T E 9/19/1982

R U N N U M B E R 1 O F 1

C U R R E N T T I M E . 3 2 3 5 E + 0 4
 S T A T I S T I C A L A R R A Y S C L E A R E D A T T I M E 0 .

* * S T A T I S T I C S F O R V A R I A B L E S B A S E D O N O B S E R V A T I O N * *

	MEAN VALUE	STANDARD DEVIATION	COEFF. OF VARIATION	MINIMUM VALUE	MAXIMUM VALUE	NO.OF OBS
EXIT INTRVL TRM1	.563E+02	.158E+03	.280E+01	.172E+01	.113E+04	295
EXIT INTRVL TRM2	.381E+03	.356E+03	.934E+00	.903E+01	.115E+04	74
EXIT INTRVL TRM3	.119E+03	.131E+03	.110E+01	.938E+01	.764E+03	238

* * F I L E S T A T I S T I C S * *

FILE NUMBER	ASSOCIATED NODE	AVERAGE LENGTH	STANDARD DEVIATION	MAXIMUM LENGTH	CURRENT LENGTH	AVERAGE WAIT TIME
1	AWAIT	0.000	0.000	1	0	0.000
2	AWAIT	.001	.045	3	0	.113
3	AWAIT	2.991	10.750	61	0	92.125
4	AWAIT	9.904	17.473	50	0	152.550
5	AWAIT	0.000	0.000	1	0	0.000
6	AWAIT	4.399	12.353	50	0	28.399
7	AWAIT	.181	1.249	10	0	6.967
8	AWAIT	9.869	13.005	30	0	570.047
9	AWAIT	.628	2.909	15	0	3.790
10		4.118	3.395	19	0	1.162

* * R E S O U R C E S T A T I S T I C S * *

RESOURCE NUMBER	RESOURCE LABEL	CURRENT CAPACITY	AVERAGE UTIL	STANDARD DEVIATION	MAXIMUM UTIL	CURRENT UTIL
1	ALFA	1	.18	.381	1	0
2	BLK1	1	.00	.052	1	0
3	BLK2	1	.10	.298	1	0
4	BETA	1	.29	.451	1	0
5	DLTA	5	.05	.294	4	0

6	PI	1	.46	.498	1	0
7	RHO	1	.03	.179	1	0
8	ZETA	1	.39	.489	1	0
9	GAMA	2	.17	.481	2	0

RESOURCE NUMBER	RESOURCE LABEL	CURRENT AVAILABLE	AVERAGE AVAILABLE	MINIMUM AVAILABLE	MAXIMUM AVAILABLE
1	ALFA	1	.8241	0	1
2	BLK1	1	.9973	0	1
3	BLK2	1	.9016	0	1
4	BETA	1	.7149	0	1
5	DLTA	5	4.9460	1	5
6	PI	1	.5389	0	1
7	RHO	1	.9668	0	1
8	ZETA	1	.6066	0	1
9	GAMA	2	1.8349	0	2

Appendix E: Test and Evaluation

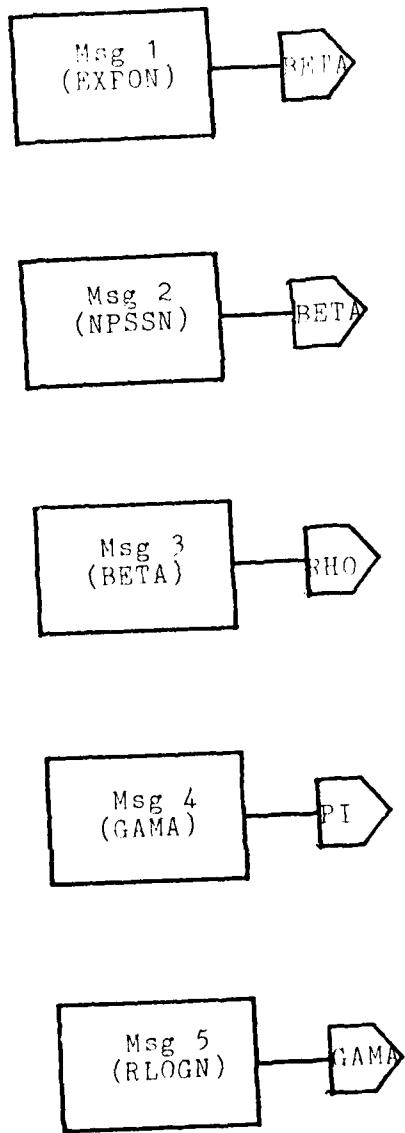
In order to check the validity of the SLAMPP, a test case was designed and evaluated. This test case was designed to use at least once every capability of SLAMPP. Specifically, nodes with balking, nodes with blocking and nodes with neither were used. Each of the nine valid distribution types found in Appendix B was used in each of their legal positions, that is, on the message creation card (MESS), process card (PROC) and the transmit card (XMIT). The default process and transmit time of zero was also demonstrated. Since balking was specified on two nodes, balk route descriptions were required in addition to regular route descriptions.

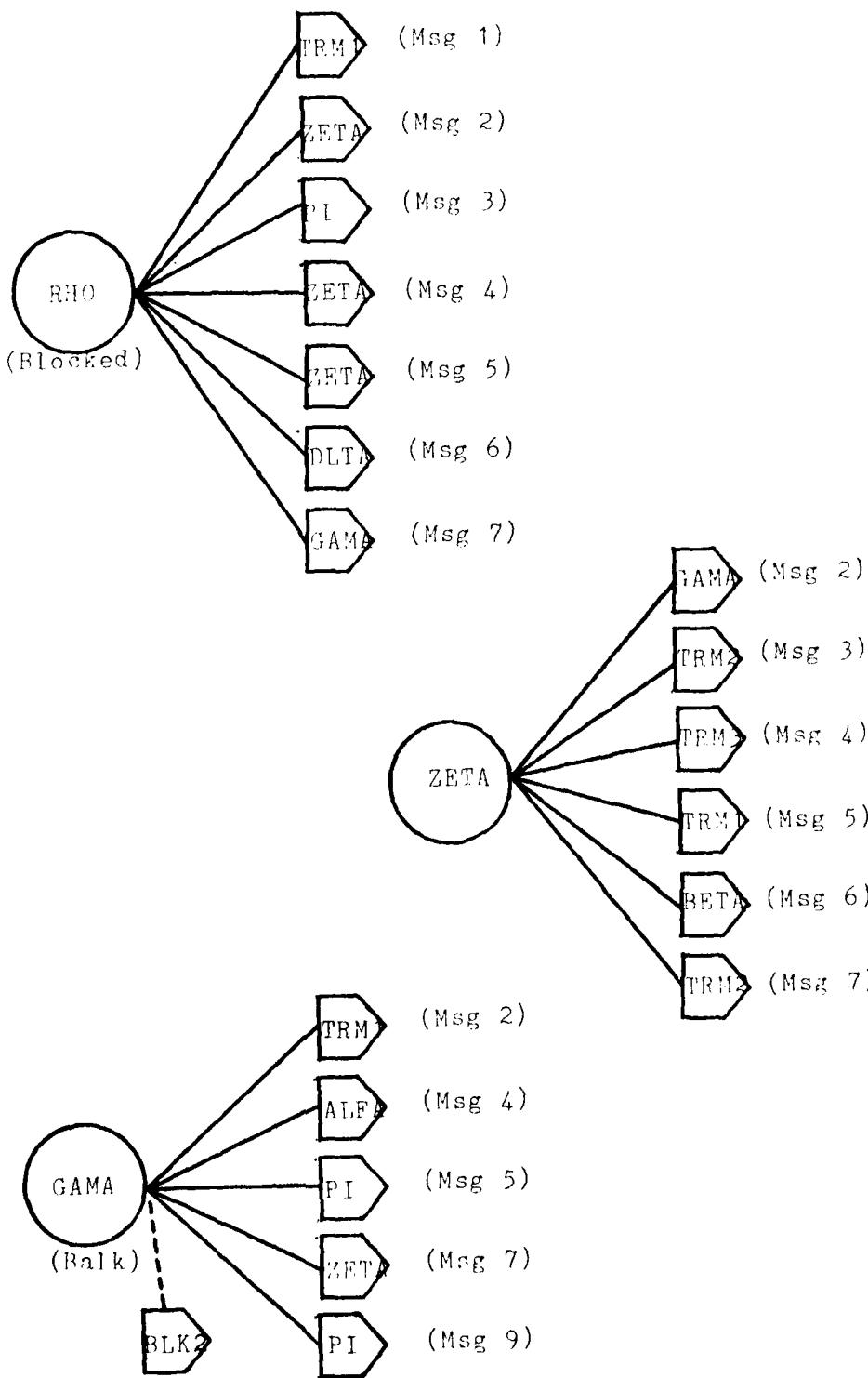
This test case was developed using the design considerations found in Appendix B. All of the nodes to be included were first determined and translated into NODE and TERM cards (a listing of the SLAMPP inputs may be found later in this appendix). The messages to be included were determined and placed into the SLAMPP input deck. Lastly, all balk routes which were necessary were supplied. The message creations and routings are diagrammatically illustrated on the following pages.

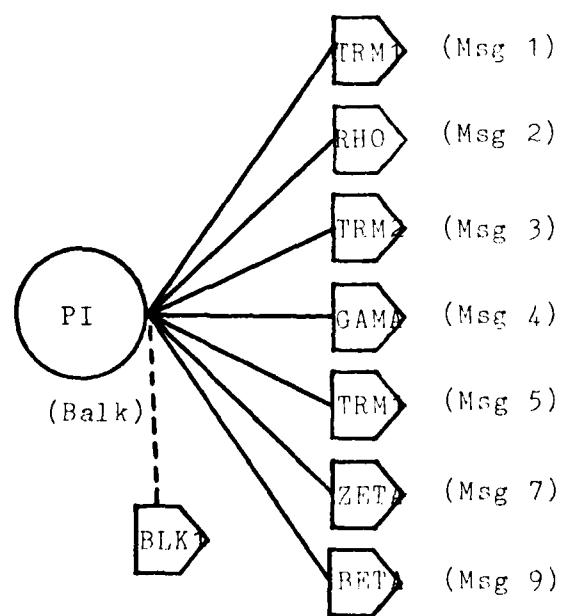
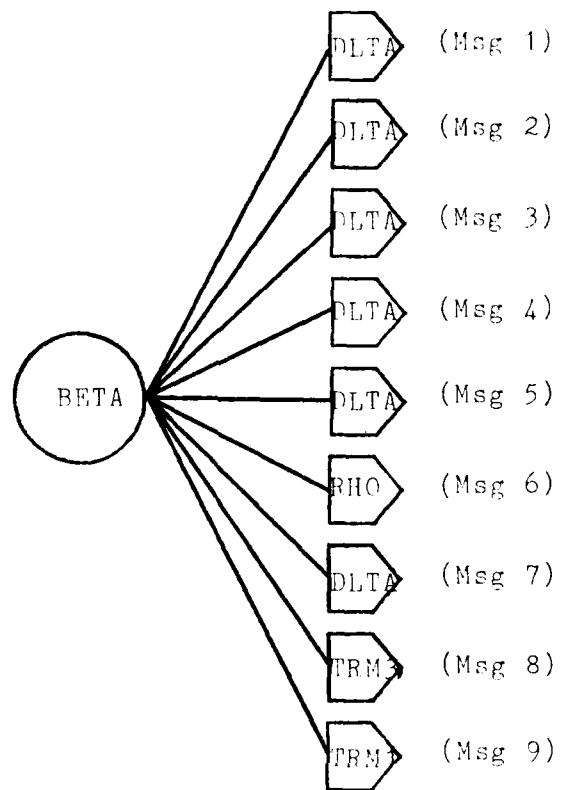
After all SLAMPP inputs were completed, the SLAMPP program was executed to produce the SLAM source code, a copy of which is at the end of this appendix. This source code was manually inspected to ensure that the source was an accurate representation of the network to be modeled. After acceptance, the source code was executed by SLAM to verify that SLAM would indeed accept the code. The SLAM execution listing is also included in this appendix.

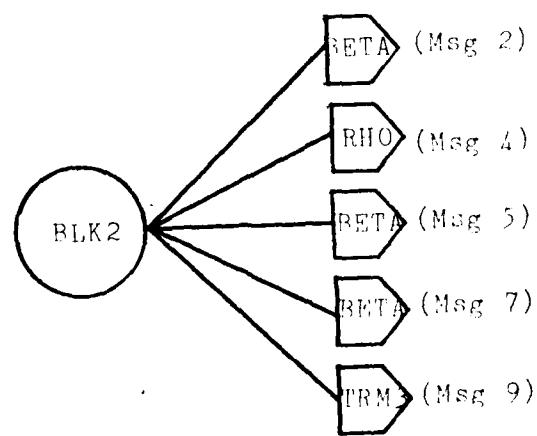
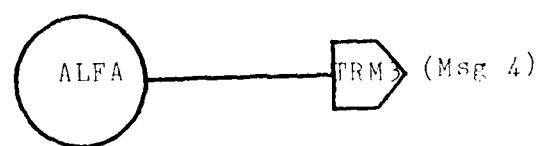
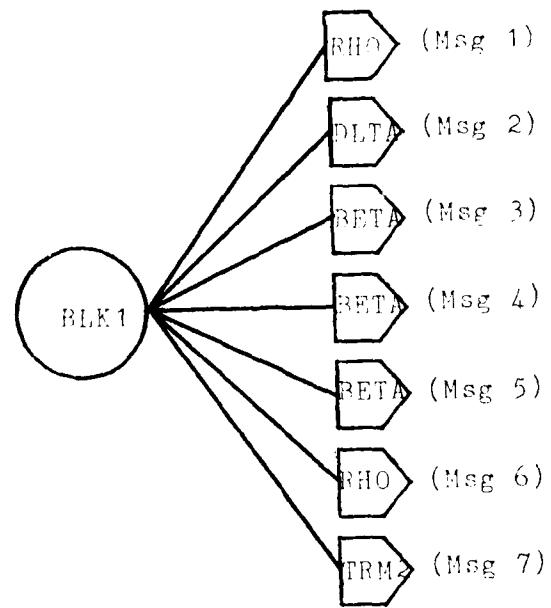
After repeated executions of SLAIIPP, no errors could be detected and the program was accepted.

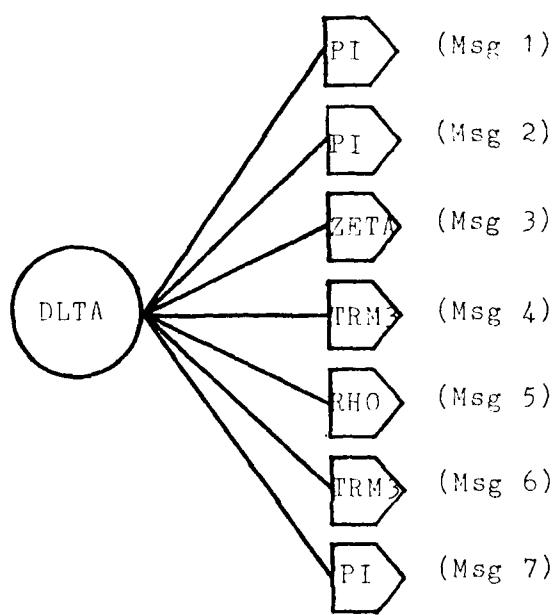
MESSAGE CREATIONS











SLAM PRE-PROCESSOR

SUMMARY OF PROGRAM INPUTS:

NODE ALFA	50.0	1.0				
NODE BLK1	9999.0	1.0				
NODE BLK2	9999.0	1.0				
NODE BETA	50.0	1.0				
NODE DLTA	20.0	5.0				
NODE PI	50.0	1.0	BLK1			
NODE RHO	10.0	1.0	BLOK			
NODE ZETA	30.0	1.0				
NODE GAMMA	15.0	2.0	BLK2			
TERM TRM 1						
TERM TRM 2						
TERM TRM 3						
MESS EXPON2.0	0.0	0.0	0.0	50.0		BETA
PROC TRIAC2.0	7.0	9.				
PATH DLTA						
XMIT CONST1.7						
PATH PI						
PROC EXPON0.75						
XMIT BETA 3.12	6.14					
PATH TRM 1						
MESS NPSSN3.2	0.0	0.0	20.0	100.0		BETA
PATH DLTA						
PROC GAMMA 2.17	.95					
XMIT CONST0.5						
PATH PI						
XMIT RLOGN1.333	.888					
PATH RHO						
PROC EXPON.23175						
XMIT NPSSN.103						
PATH ZETA						
PROC RNORM 2.1	.44					
PATH GAMMA						
PROC CONST2.0						
XMIT CONST1.1						
PATH TRM 1						
MESS BETA 2.0	1.0	0.0	10.0	75.0		RHO
PROC UNFRM 2.1	3.6					
PATH PI						
XMIT CONST6.55						
PATH TRM 2						
MESS GAMMA 6.2	5.1	0.0	0.0	100.0		PI
PROC CONST9.75						
XMIT NPSSN0.223						
PATH GAMMA						
PATH ALFA						
PROC CONST3.1						
XMIT RLOGN2.7	.334					
PATH TRM 3						

MESS RLOGN9.1	1.3	0.0	150.0	250.0	GAMA
PROC CONST0.0001					
XMIT TRIAG.85	.88	.9			
PATH PI					
PROC CONST.0001					
XMIT TRIAG.85	.88	.91			
PATH TRM1					
MESS RVORM4.1	2.6	0.0	500.0	63.0	ZETA
PATH BETA					
PROC NPSSN.667					
XMIT NPSSN.668					
PATH RHO					
XMIT GAMA .45	.23				
PATH DLTA					
PROC RNORM.382	.125				
XMIT RLOGN.543	.321				
PATH TRM3					
MESS UNFRM.75	1.5	0.0	0.0	200.0	GAMA
PATH ZETA					
PROC CONST25.0					
XMIT CONST5.0					
PATH TRM2					
MESS TRIAG2.0	5.0	10.0	15.0	125.	BETA
XMIT TRIAG3.00	6.	11.0			
PATH TRM3					
MESS CONST1.5	0.0	0.0	1.0	85.0	GAMA
PROC UNFRM4.2	6.5				
XMIT CONST1.621					
PATH PI					
PROC BETA .5	.1				
PATH BETA					
PATH TRM1					
BALK BLK1 1.0					
XMIT CONST.5					
PATH RHO					
PROC NPSSN1.6					
XMIT EXPON.0002					
PATH TRM1					
BALK BLK1 2.0					
PATH DLTA					
BALK BLK2 2.0					
XMIT GAMA 1.234	2.345				
PATH BETA					
BALK BLK1 3.0					
PROC RNORM.45	.21				
XMIT CONST2.678					
PATH BETA					
PATH DLTA					
PROC RLOGN1.2	.7654				
PATH ZETA					
XMIT BETA 6.1	3.7				
PATH TRM2					
BALK BLK1 4.0					
XMIT UNFRM.99	2.1				

PATH BETA
PATH DLTA
PROC GAMA 1.1 .87654
XMIT CONST2,3
PATH TRM 3
BALK BLK2 4.0
PATH RHO
PROC NPSSM.7
XMIT EXPON.7
PATH ZETA
PATH TRM 3
BALK BLK1 5.0
PROC CONST.00009
XMIT CONST.00099
PATH BETA
PATH DLTA
XMIT RNORM4.675 1.304
PATH RHO
XMIT RLOCH.999 .888
PATH ZETA
PROC UNFRM.54 .62
XMIT TRIAC.0003 2.5 15.0
PATH TRM 1
BALK BLK2 7.0
XMIT RNORM5.223 .162
PATH BETA
PROC TRIAC3.56 4.0 5.0
XMIT CONST.4
PATH DLTA
PATH PI
XMIT EXPON.1
PATH ZETA
BALK BLK1 7.0
PATH RHO
PATH GAMA
BALK BLK1 9.0
XMIT CONST.7
PATH TRM 2
BALK BLK2 9.0
PATH TRM 3
BALK BLK2 5.0
PATH BETA
END
NO ERRORS DETECTED. SLAM SOURCE WILL BE CONSTRUCTED.

```

1 GEN,SLAMPP,SLAM,09/19/1982,1,,,,72;
2 LIM, 9,2,500;
3 NETWORK;
4 RESOURCE/ALFA( 1), 1;
5 RESOURCE/BLK1( 1), 2;
6 RESOURCE/BLK2( 1), 3;
7 RESOURCE/BETA( 1), 4;
8 RESOURCE/DLTA( 5), 5;
9 RESOURCE/PI ( 1), 6;
10 RESOURCE/RHO ( 1), 7;
11 RESOURCE/ZETA( 1), 8;
12 RESOURCE/GAMA( 2), 9;
13 CREATE,EXPON( 2.00000), 0,1, 50;
14 ASSIGN,ATRIB(2)= 1;
15 ACT,,,BETA;
16 CREATE,NPSSN( 3.20000), 20,1, 100;
17 ASSIGN,ATRIB(2)= 2;
18 ACT,,,BETA;
19 CREATE,BETA ( 2.00000, 1.00000), 10,1, 75;
20 ASSIGN,ATRIB(2)= 3;
21 ACT,,,RHO ;
22 CREATE,GAMA ( 6.20000, 5.10000), 0,1, 100;
23 ASSIGN,ATRIB(2)= 4;
24 ACT,,,PI ;
25 CREATE,RLOGN( 9.10000, 1.30000), 150,1, 250;
26 ASSIGN,ATRIB(2)= 5;
27 ACT,,,GAMA;
28 CREATE,RNORM( 4.10000, 2.60000), 500,1, 63;
29 ASSIGN,ATRIB(2)= 6;
30 ACT,,,ZETA;
31 CREATE,UNFRM( .75000, 1.50000), 0,1, 200;
32 ASSIGN,ATRIB(2)= 7;
33 ACT,,,GAMA;
34 CREATE,TRIAG( 2.00000, 5.00000, 10.00000), 15,1,
35 ASSIGN,ATRIB(2)= 8;
36 ACT,,,BETA;
37 CREATE, 1.50000, 1,1, 85;
38 ASSIGN,ATRIB(2)= 9;
39 ACT,,,GAMA;
40 ALFA AWAIT( 1/ 50),ALFA;
41 COON,1;
42 ACT,,ATRIB(2).EQ. 4,L101;
43 L101 GOON;
44 ACT, 3.10000;
45 GOON;
46 ACT,RLOGN( 2.70000, .33400);
47 FREE,ALFA;
48 ACT,,,TRM3;
49 BLK1 AWAIT( 2/9999),BLK1;
50 COON,1;
51 ACT,,ATRIB(2).EQ. 1,L102;
52 ACT,,ATRIB(2).EQ. 2,L103;
53 ACT,,ATRIB(2).EQ. 3,L104;

```

```

54      ACT,,ATRIB(2).EQ. 4,L105;
55      ACT,,ATRIB(2).EQ. 5,L106;
56      ACT,,ATRIB(2).EQ. 7,L107;
57      ACT,,ATRIB(2).EQ. 9,L108;
58  L102  GOON;
59      ACT,.50000;
60      FREE,BLK1;
61      ACT,,,RHO;
62  L103  GOON;
63      FREE,BLK1;
64      ACT,,,DLTA;
65  L104  GOON;
66      ACT,RNORM(.45000,.21000);
67      GOON;
68      ACT,2.67800;
69      FREE,BLK1;
70      ACT,,,BETA;
71  L105  GOON;
72      ACT,UNFRM(.99000,2.10000);
73      FREE,BLK1;
74      ACT,,,BETA;
75  L106  GOON;
76      ACT,.00090;
77      GOON;
78      ACT,.00099;
79      FREE,BLK1;
80      ACT,,,BETA;
81  L107  GOON;
82      FREE,BLK1;
83      ACT,,,RHO;
84  L108  GOON;
85      ACT,.70000;
86      FREE,BLK1;
87      ACT,,,TRM2;
88  BLK2  AWAIT(3/9999),BLK2;
89      GOON,1;
90      ACT,,ATRIB(2).EQ. 2,L109;
91      ACT,,ATRIB(2).EQ. 4,L110;
92      ACT,,ATRIB(2).EQ. 5,L111;
93      ACT,,ATRIB(2).EQ. 7,L112;
94      ACT,,ATRIB(2).EQ. 9,L113;
95  L109  GOON;
96      ACT,CAMA(1.23400,2.34500);
97      FREE,BLK2;
98      ACT,,,BETA;
99  L110  GOON;
100     FREE,BLK2;
101     ACT,,,RHO;
102  L111  GOON;
103     FREE,BLK2;
104     ACT,,,BETA;
105  L112  GOON;
106     ACT,PNORM(5.22300,.16200);
107     FREE,BLK2;

```

```

108      ACT,,,BETA;
109      L113  GOON;
110      FREE,BLK2;
111      ACT,,,TRM3;
112      BETA  AWAIT( 4/ 50),BETA;
113      GOON,1;
114      ACT,,ATRIB(2).EQ. 1,L114;
115      ACT,,ATRIB(2).EQ. 2,L115;
116      ACT,,ATRIB(2).EQ. 3,L116;
117      ACT,,ATRIB(2).EQ. 4,L117;
118      ACT,,ATRIB(2).EQ. 5,L118;
119      ACT,,ATRIB(2).EQ. 6,L119;
120      ACT,,ATRIB(2).EQ. 7,L120;
121      ACT,,ATRIB(2).EQ. 8,L121;
122      ACT,,ATRIB(2).EQ. 9,L122;
123      L114  GOON;
124      ACT,TRIAG( 2.00000, 7.00000, 9.00000);
125      GOON;
126      FREE,BETA;
127      ACT,,,DLTA;
128      L115  GOON;
129      FREE,BETA;
130      ACT,,,DLTA;
131      L116  GOON;
132      FREE,BETA;
133      ACT,,,DLTA;
134      L117  GOON;
135      FREE,BETA;
136      ACT,,,DLTA;
137      L118  GOON;
138      FREE,BETA;
139      ACT,,,DLTA;
140      L119  GOON;
141      ACT,NPSSN(.66700);
142      GOON;
143      ACT,NPSSN(.66800);
144      FREE,BETA;
145      ACT,,,RHO ;
146      L120  GOON;
147      ACT,TRIAG( 3.56000, 4.00000, 5.00000);
148      GOON;
149      ACT,.40000;
150      FREE,BETA;
151      ACT,,,DLTA;
152      L121  GOON;
153      ACT,TRIAG( 3.00000, 6.00000, 11.00000);
154      FREE,BETA;
155      ACT,,,TRM3;
156      L122  GOON;
157      FREE,BETA;
158      ACT,,,TRM1;
159      DLTA  AWAIT( 5/ 20),DLTA;
160      GOON,1;
161      ACT,,ATRIB(2).EQ. 1,L123;

```

```

162      ACT,,ATRIB(2).EQ. 2,L124;
163      ACT,,ATRIB(2).EQ. 3,L125;
164      ACT,,ATRIB(2).EQ. 4,L126;
165      ACT,,ATRIB(2).EQ. 5,L127;
166      ACT,,ATRIB(2).EQ. 6,L128;
167      ACT,,ATRIB(2).EQ. 7,L129;
168  L123  GOON;
169      ACT, 1.70000;
170      FREE,DLTA;
171      ACT,,,PI  ;
172  L124  GOON;
173      ACT,GAMA ( 2.17000, .95000);
174      GOON;
175      ACT, .50000;
176      FREE,DLTA;
177      ACT,,,PI  ;
178  L125  GOON;
179      ACT,RLOGN( 1.20000, .76540);
180      GOON;
181      FREE,DLTA;
182      ACT,,,ZETA;
183  L126  GOON;
184      ACT,GAMA ( 1.10000, .87654);
185      GOON;
186      ACT, 2.30000;
187      FREE,DLTA;
188      ACT,,,TRM3;
189  L127  GOON;
190      ACT,RNORM( 4.67500, 1.30400);
191      FREE,DLTA;
192      ACT,,,RHO ;
193  L128  GOON;
194      ACT,RNORM( .38200, .12500);
195      GOON;
196      ACT,RLOGN( .54300, .32100);
197      FREE,DLTA;
198      ACT,,,TRM3;
199  L129  GOON;
200      FREE,DLTA;
201      ACT,,,PI  ;
202  PI    AWAIT( 6/ 50),PI ,BALK(BLK1);
203      GOON,1;
204      ACT,,ATRIB(2).EQ. 1,L130;
205      ACT,,ATRIB(2).EQ. 2,L131;
206      ACT,,ATRIB(2).EQ. 3,L132;
207      ACT,,ATRIB(2).EQ. 4,L133;
208      ACT,,ATRIB(2).EQ. 5,L134;
209      ACT,,ATRIB(2).EQ. 7,L135;
210      ACT,,ATRIB(2).EQ. 9,L136;
211  L130  GOON;
212      ACT,EXPON( .75000);
213      GOON;
214      ACT,BETA ( 3.12000, 6.14000);
215      FREE,PI  ;

```

```

216      ACT,,,TRM1;
217      L131  GOON;
218      ACT,PLOGN( 1.33300, .88800);
219      FREE,PI ;
220      ACT,,,RHO ;
221      L132  GOON;
222      ACT, 6.55000;
223      FREE,PI ;
224      ACT,,,TRM2;
225      L133  GOON;
226      ACT, 9.75000;
227      GOON;
228      ACT,NPSSN( .22300);
229      FREE,PI ;
230      ACT,,,GAMA;
231      L134  GOON;
232      ACT, .00010;
233      GOON;
234      ACT,TRIAG( .85000, .88000, .91000);
235      FREE,PI ;
236      ACT,,,TRM1;
237      L135  GOON;
238      ACT,EXPON( .10000);
239      FREE,PI ;
240      ACT,,,ZETA;
241      L136  GOON;
242      ACT,BETA( .50000, .10000);
243      GOON;
244      FREE,PI ;
245      ACT,,,BETA;
246      RHO   AWAIT( 7/ 10),RHO ,BLOCK;
247      GOON,1;
248      ACT,,ATRIB(2).EQ. 1,L137;
249      ACT,,ATRIB(2).EQ. 2,L138;
250      ACT,,ATRIB(2).EQ. 3,L139;
251      ACT,,ATRIB(2).EQ. 4,L140;
252      ACT,,ATRIB(2).EQ. 5,L141;
253      ACT,,ATRIB(2).EQ. 6,L142;
254      ACT,,ATRIB(2).EQ. 7,L143;
255      L137  GOON;
256      ACT,NPSSN( 1.60000);
257      GOON;
258      ACT,EXPON( .00020);
259      FREE,RHO ;
260      ACT,,,TRM1;
261      L138  GOON;
262      ACT,EXPON( .23175);
263      GOON;
264      ACT,NPSSN( .10300);
265      FREE,RHO ;
266      ACT,,,ZETA;
267      L139  GOON;
268      ACT,UNFRM( 2.10000, 3.60000);
269      GOON;

```

```

270      FREE,RHO ;
271      ACT,,,PI ;
272      L140  GOON;
273      ACT,NPSSN(   .70000);
274      GOON;
275      ACT,EXPON(   .70000);
276      FREE,RHO ;
277      ACT,,,ZETA;
278      L141  GOON;
279      ACT,RLOGN(   .99900,     .88800);
280      FREE,RHO ;
281      ACT,,,ZETA;
282      L142  GOON;
283      ACT,GAMA(   .45000,     .23000);
284      FREE,RHO ;
285      ACT,,,DLTA;
286      L143  GOON;
287      FREE,RHO ;
288      ACT,,,GAMA;
289      ZETA   AWAIT( 8/ 30),ZETA;
290      GOON,1;
291      ACT,,ATRIB(2).EQ.  2,L144;
292      ACT,,ATRIB(2).EQ.  3,L145;
293      ACT,,ATRIB(2).EQ.  4,L146;
294      ACT,,ATRIB(2).EQ.  5,L147;
295      ACT,,ATRIB(2).EQ.  6,L148;
296      ACT,,ATRIB(2).EQ.  7,L149;
297      L144  GOON;
298      ACT,RNORM(   2.10000,     .44000);
299      GOON;
300      FREE,ZETA;
301      ACT,,,GAMA;
302      L145  GOON;
303      ACT,BETA(   6.10000,     3.70000);
304      FREE,ZETA;
305      ACT,,,TRM2;
306      L146  GOON;
307      FREE,ZETA;
308      ACT,,,TRM3;
309      L147  GOON;
310      ACT,UNFRM(   .54000,     .62000);
311      GOON;
312      ACT,TRIAG(   .00030,     2.50000,   15.00000);
313      FREE,ZETA;
314      ACT,,,TRM1;
315      L148  GOON;
316      FREE,ZETA;
317      ACT,,,BETA;
318      L149  GOON;
319      ACT,   25.00000;
320      GOON;
321      ACT,   5.00000;
322      FREE,ZETA;
323      ACT,,,TRM2;

```

```
324 GAMA  AVAIT( 9/ 15 ),GAMA,BALK(BLK2);  
325      GOON,1;  
326      ACT,,ATRIB(2).EQ. 2,L150;  
327      ACT,,ATRIB(2).EQ. 4,L151;  
328      ACT,,ATRIB(2).EQ. 5,L152;  
329      ACT,,ATRIB(2).EQ. 7,L153;  
330      ACT,,ATRIB(2).EQ. 9,L154;  
331 L150  GOON;  
332      ACT, 2.00000;  
333      GOON;  
334      ACT, 1.10000;  
335      FREE,GAMA;  
336      ACT,,,TRM1;  
337 L151  GOON;  
338      FREE,GAMA;  
339      ACT,,,ALFA;  
340 L152  GOON;  
341      ACT, .00010;  
342      GOON;  
343      ACT,TRIAC( .85000, .88000, .90000);  
344      FREE,GAMA;  
345      ACT,,,PI ;  
346 L153  GOON;  
347      FREE,GAMA;  
348      ACT,,,ZETA;  
349 L154  GOON;  
350      ACT,UNFRM( 4.20000, 6.50000);  
351      GOON;  
352      ACT, 1.62100;  
353      FREE,GAMA;  
354      ACT,,,PI ;  
355 TRM1  COLCT,INT(1),EXIT INTRVL TRM1;  
356      TERM;  
357 TRM2  COLCT,INT(1),EXIT INTRVL TRM2;  
358      TERM;  
359 TRM3  COLCT,INT(1),EXIT INTRVL TRM3;  
360      TERM;  
361      END;  
362 INIT,0;  
363 FIN;
```

S L A M E C H O R E P O R T

SIMULATION PROJECT SLAM

BY SLAMPP

DATE 9/19/1982

RUN NUMBER 1 OF 1

SLAM VERSION JAN 79

GENERAL OPTIONS

PRINT INPUT STATEMENTS (ILIST):	YES
PRINT ECHO REPORT (IECHO):	YES
EXECUTE SIMULATIONS (IXQT):	YES
PRINT INTERMEDIATE RESULTS HEADING (IPIRH):	YES
PRINT SUMMARY REPORT (ISMRY):	YES

LIMITS ON FILES

MAXIMUM NUMBER OF USER FILES (MFILS):	9
MAXIMUM NUMBER OF USER ATTRIBUTES (MATR):	2
MAXIMUM NUMBER OF CONCURRENT ENTRIES (MNTRY):	500

FILE SUMMARY

FILE NUMBER	INITIAL ENTRIES	RANKING CRITERION
1	0	FIFO
2	0	FIFO
3	0	FIFO
4	0	FIFO
5	0	FIFO
6	0	FIFO
7	0	FIFO
8	0	FIFO
9	0	FIFO

STATISTICS BASED ON OBSERVATIONS

COLCT NUMBER	COLLECTION MODE	IDENTIFIER	HISTOGRAM SPECIFICATIONS
			NCEL HLOW HWID
1	NETWORK	EXIT INTRVL TRM 1	
2	NETWORK	EXIT INTRVL TRM 2	
3	NETWORK	EXIT INTRVL TRM 3	

RANDOM NUMBER STREAMS

STREAM NUMBER	SEED VALUE	REINITIALIZATION OF STREAM
1	124397822910957	NO
2	3467133063389	NO
3	79654468614391	NO
4	184170232136813	NO
5	280033029935085	NO
6	147959512963949	NO
7	125894583854829	NO
8	150477775663725	NO
9	227874746727917	NO
10	82174077946221	NO

INITIALIZATION OPTIONS

BEGINNING TIME OF SIMULATION (TTBEG):	0.
ENDING TIME OF SIMULATION (TTFIN):	.1000E+21
STATISTICAL ARRAYS CLEARED (JJCLR):	YES
VARIABLES INITIALIZED (JJVAR):	YES
FILES INITIALIZED (JJFIL):	YES

NSET/QSET STORAGE ALLOCATION

DIMENSION OF NSET/QSET (NNSET):	7000
WORDS ALLOCATED TO FILING SYSTEM:	3000
WORDS ALLOCATED TO INDEXED LIST TAGS:	0
WORDS ALLOCATED TO NETWORK:	3042
WORDS AVAILABLE FOR PLOTS/TABLES:	958

INPUT ERRORS DETECTED: 0

EXECUTION WILL BE ATTEMPTED

SLAM SUMMARY REPORT

SIMULATION PROJECT SLAM

BY SLAMPP

DATE 9/19/1982

RUN NUMBER 1 OF 1

CURRENT TIME .3235E+04
 STATISTICAL ARRAYS CLEARED AT TIME 0.

STATISTICS FOR VARIABLES BASED ON OBSERVATION

	MEAN VALUE	STANDARD DEVIATION	COEFF. OF VARIATION	MINIMUM VALUE	MAXIMUM VALUE	NO.OF OBS
EXIT INTRVL TRM1	.563E+03	.158E+03	.280E+01	.172E+01	.113E+04	295
EXIT INTRVL TRM2	.381E+03	.356E+03	.934E+00	.903E+01	.115E+04	74
EXIT INTRVL TRM3	.119E+03	.131E+03	.110E+01	.938E+01	.764E+03	238

FILE STATISTICS

FILE NUMBER	ASSOCIATED NODE	AVERAGE LENGTH	STANDARD DEVIATION	MAXIMUM LENGTH	CURRENT LENGTH	AVERAGE WAIT TIME
1	AWAIT	0.000	0.000	1	0	0.000
2	AWAIT	.001	.045	3	0	.113
3	AWAIT	2.991	10.750	61	0	92.125
4	AWAIT	9.904	17.473	50	0	152.550
5	AWAIT	0.000	0.000	1	0	0.000
6	AWAIT	4.399	12.353	50	0	28.399
7	AWAIT	.181	1.249	10	0	6.967
8	AWAIT	9.869	13.005	30	0	570.047
9	AWAIT	.628	2.909	15	0	3.790
10		4.118	3.395	19	0	1.162

RESOURCE STATISTICS

RESOURCE NUMBER	RESOURCE LABEL	CURRENT CAPACITY	AVERAGE UTIL	STANDARD DEVIATION	MAXIMUM UTIL	CURRENT UTIL
1	ALFA	1	.18	.381	1	0
2	BLK1	1	.00	.052	1	0
3	BLK2	1	.10	.298	1	0
4	BETA	1	.29	.451	1	0
5	DLTA	5	.05	.294	4	0

AD-A124 789

TACTICAL AIR CONTROL SYSTEM SIMULATION PROGRAM(U) AIR
FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF
ENGINEERING D P MCCANLESS DEC 82 AFIT/OCS/EE/82D-24

F/G 9/2

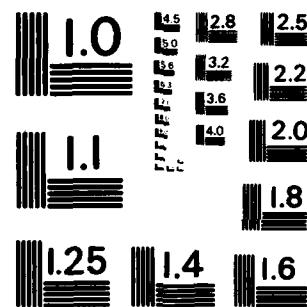
22

NL

UNCLASSIFIED



END
DATE
PRINTED
3 83
DTG



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

6	PI	1	.46	.498	1	0
7	RHO	1	.03	.179	1	0
8	ZETA	1	.39	.489	1	0
9	GAMA	2	.17	.481	2	0

RESOURCE NUMBER	RESOURCE LABEL	CURRENT AVAILABLE	AVERAGE AVAILABLE	MINIMUM AVAILABLE	MAXIMUM AVAILABLE
1	ALFA	1	.8241	0	1
2	BLK1	1	.9973	0	1
3	BLK2	1	.9016	0	1
4	BETA	1	.7149	0	1
5	DLTA	5	4.9460	1	5
6	PI	1	.5389	0	1
7	RHO	1	.9668	0	1
8	ZETA	1	.6066	0	1
9	GAMA	2	1.8349	0	2

VITA

Donald Paul McCanless was born on 31 March 1954 in San Antonio, Texas. He graduated from Highlands High School in 1972 and attended Trinity University where he received a Bachelor of Science degree in computer science in 1976. Following his commission in the United States Air Force from Officers Training School in August, 1977, he served as a programmer/analyst at Headquarters, Strategic Air Command at Offutt Air Force Base, Nebraska. In May, 1981, he entered the School of Engineering, Air Force Institute of Technology at Wright Patterson Air Force Base, Ohio.

Permanent address: 453 Menlo Blvd.
San Antonio, Texas 78223

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/82D-24	2. GOVT ACCESSION NO. AD-A124789	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) TACTICAL AIR CONTROL SYSTEM SIMULATION PROGRAM	5. TYPE OF REPORT & PERIOD COVERED MS. THESIS	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Donald P. McCauley, Captain, USAF	8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS AIR FORCE INSTITUTE OF TECHNOLOGY DEPARTMENT OF ELECTRICAL ENGINEERING WRIGHT PATTERSON AFB, OH 45433	10. PROGRAM ELEMENT, PROJECT, TASK & WORK UNIT NUMBERS	
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE December 1982	13. NUMBER OF PAGES 97
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified	
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Approved for public release; IAW AFM 100-17. Lynn E. WOLAYER Dean for Research and Professional Development Air Force Institute of Technology (ATC) Wright-Patterson AFB OH 45433	4 JAN 1983	
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computerized Simulation SLAM Communications Networks		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Tactical Air Force Interoperability Group wants to be able to simulate communication networks. The simulation language for alternative modeling (SLAM) was proposed as the basis for the simulation. However, this language requires too much effort to make any changes to the network to be simulated. A Pre-Processor was written in FORTRAN 77 which allows the user to describe the network in a convenient manner, and produces the necessary SLAM source code to perform the simulation.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

END
DATE
FILMED

3-83

DTI